A FUNCTIONAL MODELING FRAMEWORK FOR INTERDISCIPLINARY BUILDING DESIGN

A Thesis
Presented to
The Academic Faculty

by

Andrés Cavieres

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Architecture

Georgia Institute of Technology
August, 2018

COPYRIGHT© 2018 BY ANDRÉS CAVIERES

A FUNCTIONAL MODELING FRAMEWORK FOR INTERDISCIPLINARY BUILDING DESIGN

Approved by:

Professor Charles Eastman, Advisor
School of Architecture
Georgia Institute of Technology

Professor Russell Gentry, Co-Advisor
School of Architecture
Georgia Institute of Technology

Professor Godfried Augenbroe
School of Architecture
Georgia Institute of Technology

Professor Ashok Goel
School of Interactive Computing
Georgia Institute of Technology

Dr. Stefano Borgo
Laboratory for Applied Ontology
ICST-CNR

Dr. Mehdi Nourbakhsh
Autodesk Research
Autodesk

Date Approved: July, 24th 2018

"Sometimes something's got to happen before something is going to happen."

Johan Cruyff.

To my beloved wife Aglaia, the best gift of my life.

ACKNOWLEDGEMENTS

The work presented in this dissertation would not have been possible without the support and contribution of many people. I am especially grateful to my advisor, Prof. Chuck Eastman for all his guidance, encouragement and patience throughout these years. His wisdom and intellectual rigor have inspired me to continue in the pursuit of excellence and relevance, despite the challenges. It has been truly a privilege to be one of his students. I also have been extremely fortunate of having Prof. Russell Gentry as my co-advisor and mentor. His patience, generosity and optimism, always making himself available, have been crucial to overcome the most difficult moments. Many of my accomplishments were only possible thanks to his continuous support and friendship.

I also have been fortunate enough to have an outstanding dissertation committee. Some of the most important intellectual contributions to this research come directly from the large body of scholarly work of its members. I am especially grateful to Prof. Godfried Augenbroe, who first introduced me to the concept of Aspect Systems, which is so fundamental to this research. When I finally understood the central idea, a whole new perspective on the purpose of design was opened in front of me. That realization led me to pursue a minor in Artificial Intelligence with Prof. Ashok Goel; a path that proved to be both difficult and rewarding. His extensive body of work in Knowledge Representation and Functional Modeling was key to start shaping the theoretical framework of this dissertation, and his feedback during the process allowed me to fill many theoretical gaps that otherwise would remain unnoticed. I would like to specially acknowledge the contribution of Dr. Stefano Borgo, from the Laboratory of Applied Ontology, at the Institute of Cognitive Sciences and Technologies (ISTC) in Italy. His work in applied ontology, and particularly in the formalization of the multiple meanings of function have been pivotal for this dissertation. I am very grateful for his continuous support from the very first day that I wrote him. This dissertation became significantly better thanks to his substantive criticism and constructive

feedback. Finally, I would like to express my gratitude to Dr. Mehdi Nourbakhsh, Senior Researcher at Autodesk. His insights regarding the practical aspects of implementation and usability of future functional modeling applications have been important to frame the scope and complexity of the model proposed. While not totally resolved, due to the early stage of this research, a foundation has been developed based on his feedback.

I am also grateful to my colleagues at the Ph.D. program at Georgia Institute of Technology, with whom I share many good moments. Special thanks to Hugo Sheward, Hui Cai, Jin-Kook Lee, Dong-Hoon Yang, Julie Zook, Marcelo Bernal, Paula Gomez, Shani Sharif and Pedro Soza. I also would like to acknowledge Francisco Valdes, and our collaboration at the Digital Fabrication Lab, from which many new ideas emerged. I would like to especially thank my colleague and friend Tristan Al-Haddad, whose friendship made my time at Georgia Tech significantly better. My gratitude also extends to Mr. Karl Brohammer, the ex-director Digital Fabrication Lab; and to Robin Tucker, for their dedication, professionalism and care.

I also would like to express my most sincere gratitude to Dr. Stephanie Pilat, Chair of the Division of Architecture, and Prof. Hans Butzer, Dean of the Christopher C. Gibbs College of Architecture at the University of Oklahoma, for their continuous support during these last stages of my dissertation.

This journey started many years ago, even before I considered the possibility of pursuing doctoral studies. The credit for such decision, and for helping to make it a reality, is to Dr. Eduardo Lyon, who was my professor and first mentor at Universidad de Chile. Many of the early ideas underpinning this research come from that period, and the collaboration with my friend Gonzalo Vargas.

Finally, I cannot thank enough my family for all the love, support and dedication given to me during this entire process. This dissertation is a tribute to all of you. In particular, I dedicate this dissertation to my wife Aglaia de Biagi, to my son Felipe Cavieres, my sister Pamela Cavieres, and to my parents, Blancaluz Pinilla and Patricio Cavieres.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF CODE LISTINGS

SUMMARY

The process of Building Design, as in many other forms of design, requires the effective integration of different types of knowledge. However, and in the specific context of Building Information Modeling, only structural knowledge is formally represented. Other types of necessary knowledge, such as those related to the functionality of design, and the set of causal behaviors from which such functionality is delivered, remain tacit or indirectly referenced by using structural properties as proxy representations (e.g. geometry). The lack of a more comprehensive and rigorous representational framework to formally describe various behavioral and functional aspects of buildings limits the scope of semantics required to support more effective interdisciplinary collaboration and design integration. In particular, there is a lack of computational support to describe cross-cutting behavioral interactions and side-effects that occur among different building sub-systems, which often play a role in the satisfaction of functional goals.

To address this problem, the research proposes the development of a representational framework for the functional and behavioral characterization of building systems and components based on the Functional Representation (FR) schema developed by Chandrasekaran and Josephson (2000), and its recent formalization following the DOLCE foundation ontology, by Borgo et al. (2009). A subset of FR axioms has been translated into Description Logic using the Web Ontology Language (OWL-DL) to explore query capabilities of the proposed framework to support identification of behavioral interactions based on inference capabilities of available OWL-DL reasoners.

The dissertation provides a theoretical basis for the formulation of functional modeling capabilities currently not available in Building Design. In particular, these capabilities are intended to support the incremental elucidation of behavioral interactions that emerge across different building sub-systems, based on the principle of co-participation of structural entities in a same behavioral phenomena (category of perdurants). The elucidation is expected to be supported by computational inference from structural relations asserted in BIM models by various stakeholders, and at different stages of the design process.

CHAPTER I

INTRODUCTION

"We think we do not have knowledge of a thing until we have grasped its why, that is to say, its cause." Aristotle (Phys. 194 b17-20)

The process of design is largely predicated in the ability of designers to articulate the diversity of forces acting, sometimes antagonistically, at various levels of a design problem. Often, this effort involves the creative transformation of what is seen initially as limitations and constraints into opportunities to achieve better, more intelligent, and often more beautiful solutions. This type of sophistication is key in situations where resources are scarce and extreme pragmatism is sometimes a matter of survival.

Many examples from the past exhibit such combination of resourcefulness and ingenuity. In the past however, design problems were usually simpler and remained relatively the same over long periods of time. According to Alexander, the number of forces and force interactions designers had to deal with were small and for all practical purposes, well understood. This means that individual engineers and architects could learn by means of apprenticeship and practice everything that was necessary to design fully functional mechanical devices, civil structures and buildings. Trial and error was not only accepted as design method, but the only method available to explore design alternatives and to slowly advance new design knowledge. By contrast, the increasing complexity of contemporary society, and its ever growing set of technological and environmental needs does not allow the same approach. In fact, new design solutions have to be developed at much faster pace for problems that did not even exist before [9].

To cope with this new level of complexity, many design problems started to be addressed from a multidisciplinary perspective. A general approach adopted in such scenarios was the divide-and-conquer method of breaking down complexity into a number of smaller and more manageable aspects of the general problem. Each of these aspects are, generally

speaking, discipline-specific, and therefore are expected to be handled by domain experts in relative isolation from others. Yet, the final integration of these seemingly disparate aspects has become a major challenge [111, 128]. While partial solutions may be found for individual sub-systems within the scope of discrete disciplines, conflicting interactions do emerge frequently at the boundaries between sub-systems.

From a multidisciplinary perspective, the critical problem is how to collaboratively identify and manage such conflicting interactions in a timely and cost-effective manner, so that the final purpose of the design is achieved with minimal degrees of uncertainty and risk. These considerations suggest the need for a representational framework that allows the description of relevant behavioral interactions in a way that is more transparent and commonsensical than specialized, domain-specific models. Notice that, while individual forces that create the interactions may be domain-specific, their interactions are usually not. In a very literal sense, these interactions and side-effects are indeed inter-disciplinary.

Unfortunately, such an "in-between" dimension of design is lacking an explicit, formal representation. For this reason, much of the information that is crucial for interdisciplinary collaboration is treated during the design process in a tacit or ambiguous manner. Thus, non-stated assumptions and wishful thinking dominates much of the decision-making regarding how different systems would "work together" in reality. This attitude is certainly compensatory for the absence of more effective means to describe, communicate, reason about, and ultimately, to act upon systems integration in collaborative terms. In the absence of such means, uncertainty and risk are often underestimated, which ultimately plays against any stated expectation of performance. [16].

1.1 Teleological consistency

In order to support interdisciplinary design collaboration, a common, formal representational framework is needed to describe the classes of interactions that are relevant across parts of a system. Such framework has to enable the construction of models that are meaningful within and across the disciplines involved. But, what kind of models are these? And from a more philosophical perspective, which types of real-world entities should these

models commit to?

In the scope of this research, the first question refers to the problem of defining machine-readable models of causality, within and across systems, and at multiple levels of abstraction. The second question refers to the problem of ontology and the specific types of ontological commitments that these models have to share among them. Since the required levels of consistency among different domain models can only be achieved if they refer, albeit differently, to the same type of real-world entities, then it seems clear that both questions are inextricably connected.

Thus, from this ontological point of view, it is not sufficient to agree on the existence of a subset of structural features acting as interface between disciplines. For example, within a CAD application, a mechanical engineer may want to increase the thickness of a plastic casing for an electronic device, in order to enhance its impact resistance. While doing so, she may not be fully aware of all the thermal and manufacturing implications of the change. The thermal model and the manufacturing model, under the control of different engineers, may still seem consistent with the mechanical CAD model since they all refer to the same thickness parameter. Yet, such consistency is purely structural (i.e. geometric). From a behavioral perspective however, requirements related to thermal and manufacturing performance may have been unintentionally compromised. It can be said that the different models of the device are no longer consistent in relation to the set of purposes they were intended to address.

This lack of semantic consistency at the teleological level denotes a general problem of representation that is common to many design domains [109, 31]. In particular, the ontological framework underlying design models are fundamentally limited, by not explicitly acknowledging the existence of multiple purposes for a design, much less different causal relationships upon which such purposes are *intended* to be achieved [151, 74, 71].

This means that, under the universe of discourse of the aforementioned mechanical CAD model, the behavioral *phenomena* associated with the thermal and manufacturing requirements simply *do not exist*. Hence, any consideration that the mechanical engineer may have about them during the design process is based solely on her own tacit knowledge

and judgment. Given a lack of expertise, or simply oversight, those important aspects may be completely ignored until very late in the process. Unfortunately, decision-making under similar circumstances is very common in practice, spanning across the spectrum of design-related disciplines. Often, changes are made without a whole-system perspective and understanding of their implications beyond the local, short-term improvement goals [118].

This representational limitation may indeed contribute to costly and sometimes completely catastrophic mistakes. In the domain of Building Design, several examples illustrate the point. For instance, the case of the collapse of two elevated walkways in the Hyatt Regency Hotel in Kansas City, caused by changes in the specification of supporting steel rods to make the walkways visually "lighter" and easier to fabricate [223]; the collapse of the Tropicana Parking Garage in Atlantic City, arguably because of late minute changes in the specification of reinforcing rods of concrete slabs and underlying beams, intended to speed up the construction process and to save money [221]; and the case of the CitiCorp Center in New York, where changes in the specification of diagonal brace connections were made to reduce costs without considering behavioral couplings with certain type of wind loads. Fortunately in this case, the structural engineer in charge of the design recognized the implications of the change soon after construction was completed and was able to convince the owners to take the necessary, albeit very expensive, corrective measures [236].

Many other similar examples of structural failure in buildings, bridges and dams caused by design decisions that did not anticipate behavioral side-effects are provided by Delatte [92]. Design changes, errors and omissions also have a significant effect in the performance and overall cost of construction activities, as discussed by Kamara et al.[193] and Hwang et al. [185]. More usually, the performance of high-level building functions such as providing thermal, acoustic and visual comfort for users, as well as indoor air quality and good ergonomics is also compromised [264]. This often play a negative role in the health and well-being of people, which in turn may have serious economic impacts derived from reduced productivity and increased employee absenteeism [10, 4].

It may be argued that an alternative approach to control interactions and side-effects

would be the definition of rules that constrain the relationship among critical design parameters, in such a way that only solutions that are valid across systems may be generated. This approach has been tried in several early implementations of Computer-Aided Design applications, especially in the domain of Building Design [109]. More recent examples are based on methods developed in Aerospace and Mechanical Engineering, under the umbrella of Multidisciplinary Design Optimization (MDO) [169, 276, 216]. There have been also attempts to use MDO in the domain of building design, particularly for optimization of building structures and energy performance [143, 130, 261]. However, as MDO focus fundamentally on parametric optimization of well understood solution spaces. This means that MDO is limited to design situations where sets of requirements are fixed, with clearly defined boundary conditions and rigid hierarchy of functional dependencies (e.g. routine design problems).

As the set of requirements involved in a routine design problem is typically well known and not subject to significant changes during the design process, key parametric relationships, including key behavioral couplings can be captured from the outset. This in turn facilitates the adoption of prescriptive design methodologies that rely heavily on the availability of predefined components and configurations. Unfortunately, this "kit-of-parts" approach which parametric optimization relies on is essentially brittle in the face of requirement changes and uncertainty normally found in novel or more complex design problems [102, 111].

In situations where innovation is important or contextual conditions change constantly, the problem space is not only ambiguous, but subject to constant redefinition as new requirements arise. In practice, this means that the problem space cannot be completely understood from the outset. Rather it tends to co-evolve with design itself [107, 227, 161, 95, 94]. Therefore, the applicability of prescriptive design rules and predefined parametric constraints is of limited use. Instead, design has to be based on a process of incremental and collaborative discovery of behavioral interactions that could lead to unanticipated effects. In particular, such process has to be driven by a continuous assessment of the potential impact that such evolving interactions may have on the satisfaction of high-level functional

goals, as stated by various stakeholders across the life-cycle of a product.

It is important to realize that behavioral couplings, interactions and side-effects do not always cause negative consequences. In many cases, design innovation may take place by taking advantage of unexpected positive interactions that sometimes occur when things are put together or used in a different way [9, 173]. In fact, positive interactions do sometimes provide opportunities for "doing more with less", if special attention is given to the way design problems are framed. In particular, such framing needs to focus on desired behavioral outcomes rather than on structural features of a problem, as it is often the case [70].

In general, current modeling tools do not facilitate this process of incremental elucidation and management of behavioral interactions. Existing approaches have to rely on intense development of custom modeling tools and methods for very specific types of design projects. These solutions demand significant technical and human resources, usually only available to large design and construction firms with internal R&D capabilities. However, for the majority of firms and companies in the AEC industry that not have access to such resources, the problem remains an important challenge.

From an interdisciplinary perspective, one of the main limitations stem from the lack of a high-level representational language to describe such interactions in formal, compatible terms. Because of that limitation, many relevant interactions and side-effects are only referred to during design by tacit or ad-hoc means. Often this role is played by geometry, which is used as representational surrogate for various behavioral aspects of a design problem. Given the informal nature of the attribution, semantic consistency across disciplines is compromised.

This limitation indicates the need for a more comprehensive modeling framework that explicitly commits to the existence of behavioral entities, including the classes of interactions and side-effects that play a role in the satisfaction of various life-cycle design functions. In order to reduce representational complexity, along with potential cognitive and computational burden, a hierarchical, modular and minimalistic approach is needed, where the semantics of relevant behaviors can be represented efficiently at successive levels of abstraction. The idea is that, in principle, the higher the level of abstraction in a hierarchy of

functions, the more general and commonsensical its meaning [3]. Such observation provides the basis for the formulation of a functional modeling framework to support interdisciplinary, multi-stakeholder collaboration in the design of socio-technical systems such as buildings.

1.2 The role of functional abstractions in interdisciplinary design

A design function is a behavioral abstraction at the top of the abstraction hierarchy [157]. Indeed, a function is the most commonsensical representation of a system behavior because it stands for its *purpose* within a context of use. At the same time, a properly formulated function also provides a reference to the set of causal mechanisms by which it may be fulfilled [76]. By carrying both meanings, a formal representation of functions is intended to support a series of design tasks. At the most general level, these can be summarized in three main groups. First, a representation of function should work as index of systems and components, necessary for reference, selection and retrieval from memory (e.g. component libraries). Second, it should help to convey the rationale for design decisions, by means of pointers to internal causal mechanisms. Third, it should indicate the means by which a design solution needs to be validated (e.g. tests and experiments). These requirements are relevant in the context of interdisciplinary collaboration, because they provide a formal basis for the identification of behavioral interactions that need to be handled collaboratively for the achievement of common goals.

In other words, the set of causal behaviors by which such a function is realized, including known side-effects, define the structure of collaboration itself. As the specification of functions and associated behaviors are subject to constant change and refinement, so does the structure of collaboration too. In particular, this implies changes in the scope and depth of interdisciplinary knowledge that needs to be brought into the design process. Such a dynamic relationship depends significantly on how design functions are represented in first place, and how flexible the representation medium is to accommodate the evolving, and often uncertain links between problem and solution spaces [227, 161].

In general, while the process of identification of interactions and side-effects may not be difficult in projects of low complexity, it is a challenge in projects characterized by a large

number of functional requirements spanning across the entire life-cycle of a product. In these types of projects, the traditional, document-centric approaches for specification of functional requirements are extremely problematic[269]. Typically, these rely on text and spreadsheet files that are not linked nor connected to the design in any formal way. Besides being time consuming and prone to error, the use of document-centric representations leads to different types of inconsistencies, especially when both requirements and design alternatives change constantly, and the consequences need to be tracked down and propagated manually across various domain models [193, 118, 175, 257]. Furthermore, document-based approaches makes it very difficult to provide any form of computational support for the management of requirements and the resolution of conflicting side-effects.

In the next subsections, two examples will be discussed to illustrate the need for smarter machine-readable representations of functionality in the context of interdisciplinary Building Design.

1.2.1 Example 1: Functional requirements for a masonry cavity wall

This case exemplifies situations when the material composition for assembly parts are changed. The change itself may be originated by unanticipated problems in the supply change, or new performance mandate from a stakeholder (e.g. client or authority with jurisdiction). For example, in the domain of masonry construction, the backup system of a brick veneer masonry wall may be changed from concrete blocks (CMU) to metal studs, arguably to speed up the on-site assembly process (Figure 1.1).

However, this change triggers an entire chain of behavioral implications that may impact negatively the acoustic and the energy performance of a building, among other problems. In the first situation, acoustic performance may be compromised because of the higher rate of sound propagation added to the wall assembly. This can be problematic in certain types of buildings, especially in multi-family housing. The second situation happens because of the potential for thermal bridging added by the metal studs. It is known that this phenomenon can reduce thermal insulation capacity of a wall up to 60% [232]. Additionally, given that metal studs are subject to a high rate of deflection, they may cause cracks in the brick

veneer, eventually leading to water infiltration, which in turn could lead to mold growth as well as corrosion of the metal studs.

In a value-driven design process where the totality of life-cycle requirements are important, these implications will need to be taken into account and resolved in a timely manner. This implies the involvement of a completely different set of domain-experts and analysis procedures that in the previous scenario where a CMU backup system was used. In theory, design changes like this should lead to a significant change in the structure of the work-flow itself. Unfortunately, such an implication is often overlooked in practice. Furthermore, it is clear that the traditional methods of documenting life-cycle requirements and design functions do not facilitate the process either. In consequence, the identification and resolution of conflicting behavioral effects is made in an ad-hoc basis, by relying almost exclusively in the judgment and tacit expertise of stakeholders involved.

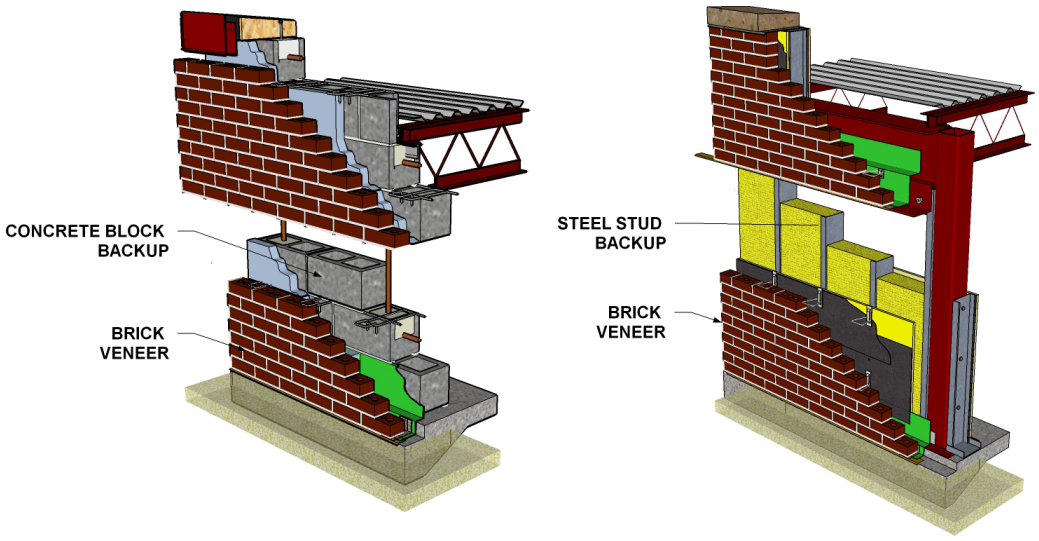

Figure 1.1: Two options for masonry cavity walls with brick veneer. On the left, a reinforced concrete block backup system. On the right, metal stud backup system.

1.2.2 Example 2: Functional requirements for a PV racking system

The second example refers to the design of a mounting device for the installation of photovoltaic systems on building rooftops, illustrated in Figure 1.2. The design was result

of a large research project funded by the U.S. Department of Energy, intended to reduce manufacturing and labor cost associated to the commercial deployment of PV systems. To achieve this goal one of the main design principles was the development of multi-functional componentry, so that cost could be reduced by significantly minimizing material use and part count. Additionally, the entire installation process needed to be tool-free to simplify and speed up the installation process [70, 67].

In the case of flat-roof buildings, the position of PV systems has been traditionally secured by using ballast instead of mechanical connections to the roof structure. This is done to avoid penetration of the roof membrane, which may void the warranty of roof vendors. Additionally, wind deflectors are attached to the back of each PV module to minimize uplift wind loads, thus contributing to the reduction of total ballast required.

One of the innovations of this design is that the wind-deflector also fulfills other important life-cycle functions. First, it works as packaging device to facilitate storage and transportation. This means that all the hardware required for the installation of a PV system can be flat-packed inside the wind deflector, thus avoiding the need for additional packaging material and their disposal after installation. Second, the wind-deflector can be used as measurement and alignment device for the proper position of rails and modules during installation. An important sub-requirement of this is the ability to support a wide variety of PV module lengths as well dimensional tolerancing. In this way, installers do not need to use tape measurement, chalk lines or similar tools that complicate the process significantly (Figure 1.2). This was considered to be a key advantage, since measuring and squaring were among the most important cost drivers during installation. Finally, the wind-deflector works as wire management device (i.e. wire trays) to hold bundles of electrical wires running under rows of adjacent PV modules away from the roof surface.

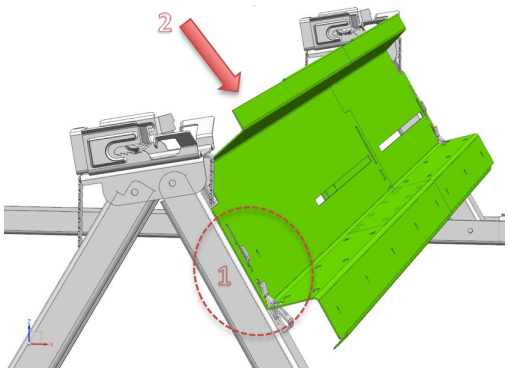

Figure 1.2: Example of a multi-functional component: PV racking wind-deflector.

At some later point during the design process, the elimination of the wind-deflector was considered, to push the manufacturing cost even lower. To compensate for uplift wind loads, it was decided to increase the sectional area of structural rails under the PV modules. Given that the new manufacturing costs were very attractive, it was estimated that the other additional wind-deflector functions such as packaging and wire-management could be sacrificed. However, the second function, measuring / squaring, was completely omitted during the decision-making process. This was one of most important features of the system, which contributed significantly to the reduction of installation time and cost. Despite its relevance, no one in the design team recalled that requirement until much later. The fact that the entire set of life-cycle requirements was specified in a central model did not facilitate the identification of the problem either. For one thing, the model was relatively large and hard to navigate. More importantly, the requirements model was incapable of raising red-flags automatically to point out the new conflict.

1.3 Towards a representation of behavioral interactions

The masonry cavity wall example illustrates situations where certain unintended behaviors are potentially introduced in the system by a change in the material of an assembly part. In other situations however the opposite may occur. As illustrated in the case of the racking wind-deflector, it might be the case that intended behavioral effects are inadvertently removed from the system, either because a material property is changed or a part is eliminated. Similarly, new interactions arise by adding new parts or changing the topological

configuration of the system (e.g. spatial layouts or circulation schemes in buildings, the assembly order of parts, etc.). The latter case will be elaborated further in the context of an office space, as part of the discussion at the end of Chapter V: Function in Artificial Intelligence (see 4). These examples will be used as case studies for the functional modeling framework implemented as proof-of-concept for validation of the research hypothesis.

It is important to note that just the possibility of negative side-effects should not prevent a change to be made, especially when such changes are intended to cover a different set of needs. It is precisely the role of different domain experts to evaluate the impact caused by a behavioral interaction. In particular, the role of domain experts is to assess the extent to which the satisfaction of functional requirements could be made *worse* or *better* by the occurrence of a given class of interactions.

Thus, a series of performance assessments developed over the full range of life-cycle interactions should provide the basis for trade-off analysis and decision-making within an interdisciplinary setting. However, this process cannot be done effectively if life-cycle functional requirements are not described in a way that is shared and transparent to all stakeholders involved. Furthermore, even if requirements are described in such a way, the collaboration may be hindered if the evolution of underlying behavioral interactions are not dynamically captured in a shared model during the process [202].

As already said before, there is currently no computational support in design in general, and much less in Building Design, for dynamically capturing the evolving functional relationships that emerge between different building subsystems due to emergent behavioral interactions. This is problematic since the achievement of performance goals, especially at higher levels of functionality are very much dependent of the nature of those interactions [264, 17].

On the other hand, research in building information modeling and building simulation has long recognized that describing all possible combinations of interactions, even at a general level, is not practical nor desirable [110, 91, 19]. Nevertheless, three observations are important to overcome these difficulties:

1. First, for any practical purpose, only a subset of behavioral interactions can really be

considered as meaningful. In particular, the relevance of any given interaction can only be established in relation to a explicitly formulated functional need, described in terms of quantifiable performance. Hence, any given behavioral interaction is meaningless if performance goals that could be potentially affected by it are not expressed and communicated in first place.

2. Second, behavioral interactions do not need to be described extensionally (i.e. by exhaustive enumeration), but can be described intensionally, by means of formal rules or conditions that describe the criteria for membership under certain patterns of interactions. In practical terms this means that, given certain formal descriptions of general patterns of interactions that tend to recur within multiple domains, more domain-specific patterns can be subsumed. The inference of subsumption requires the identification of types of structural elements that may be responsible for such interactions. Resolution of behavioral conflicts can only be resolved through the collaborative manipulation of those elements.

3. Third, there are well known patterns of interactions which tend to recur overtime for certain combinations of building system types. The cavity masonry wall example discussed previously illustrates an example of interaction pattern concerning assemblies of materials with different deflection or thermal expansion rates. Knowledge about most relevant combinations for different uses are commonly available in design guidelines and product handbooks throughout the building industry.

These observations describe in broader terms the main research problem of this dissertation. They also describe its structure and organization. In the next section the research is further characterized into general and specific problems. The research goals, methodology and expected contribution are also presented.

1.4 Research problem: Elucidation of aspect systems

The research proposes the development of a representational framework for the functional and behavioral characterization of building systems and components. A main goal of this

framework is to support the incremental elucidation of building elements (i.e. structural parts) that participate in the satisfaction of different functional requirements. In this research, the set of structural elements that participate in the satisfaction of a functional requirement is considered a special type of aggregation abstraction called an *aspect system* [16]. In other words, an aspect system is an aggregation of structural elements, possibly from two or more different subsystems, that are is meaningful from a functional perspective.

The preliminary hypothesis of this research is that structural elements having membership in two or more different aspect systems imply the possible existence of behavioral interactions among such aspect systems, and therefore, a possible conflict among different functions. However, in a first approach, there is no need for qualification of interactions, in relation to the type of role they may play in the functionality of subsystems involved, i.e. positive or negative, nor the degree of their impact, i.e. their performance. Raising a "red-flag" may be considered the simplest form of computational support, leaving the evaluation and resolution of potential conflicts to experts and stakeholders involved. More advanced capabilities are envisioned based on the qualification of behavioral interactions according to the role that they play, and the degree of their influence or impact.

1.4.1 Specific research problem

The specific research problem of this dissertation is the development of a functional modeling framework capable of supporting the inference of aspect systems. This is considered necessary for enabling more effective interdisciplinary collaboration in building systems integration under multiple performance criteria. The specific problem is divided into three incremental and complementary objectives:

- To provide the formalism for the model-based representation of functions, both from a demand perspective, concerned with the specification of needs, goals and requirements, as well as from a supply perspective, concerned with the description of functional roles played by building elements.
- To provide computational support for the elucidation of Aspects Systems, given a machine-readable representation of functions, from both the demand and the supply

perspectives.

- To provide computational support for the identification of interactions, inter-dependencies and potential conflicts between different required functions. This is based on the membership of building elements in two or more aspect systems.

A fourth objective would be the to provide computational support for the qualitative characterization of the behavioral interactions involved and their impact in the satisfaction of a functional performance requirement. This means if an interaction impacts either positively or negatively the satisfaction of a requirement, as well as the degree of the impact.

The quantitative evaluation of the impact, as part of a larger effort of performance evaluation is outside the scope of this research. However, the method for elucidation of aspect systems proposed here attempts to provide a foundation to tackle the problem of automatic generation of input models for performance analysis applications (e.g. numerical simulation).

At the implementation level, the problem involves the development of a knowledge-base model to support interdisciplinary collaboration in building design. In principle, such knowledge-based model should be able to answer the different types of query, both at class and instance levels. The ability to make inferences at both levels is a characteristic of Description Logic implemented in OWL-DL [135, 306, 209]. However, incremental elucidation of aspect systems demands reasoning about instances for the most part, given that it is during instantiation when behavioral interactions can be more easily identified.

With this consideration, a first query of interest for the elucidation of Aspect Systems is as follows:

- Given a required function f and a design model a, return the set of building elements from a that participate in the satisfaction of e.

This is the most general form of the query, for which the participation relation is not qualified in terms of positive or negative participation (i.e. the requirement is affected in a positive or negative way). A more specialized query in which the results get filtered by a qualification of the participation also needs to be formulated.

Since a performance requirement is a further specification of a function [1], for which a quantifiable measurement of satisfaction has been defined in terms of behavioral constraints [264, 16], it demands a more specific query. This would take the following form:

- Given a performance requirement f_p and a design model a, return the set of building elements from a that participate in the satisfaction of f_p.

In both cases, the returned set of building elements, possibly from different building subsystems, constitutes the aspect system of the function or performance requirement. In this way, an aspect system denotes a subset of the building system that is meaningful from a functional perspective. This functional perspective can be achieved by means of decomposition of high-level functional requirements, and needs to be agreed upon by relevant stakeholders [16]. Figure 1.3 illustrates the process of functional decomposition and the specialization of functions in terms of performance requirements.

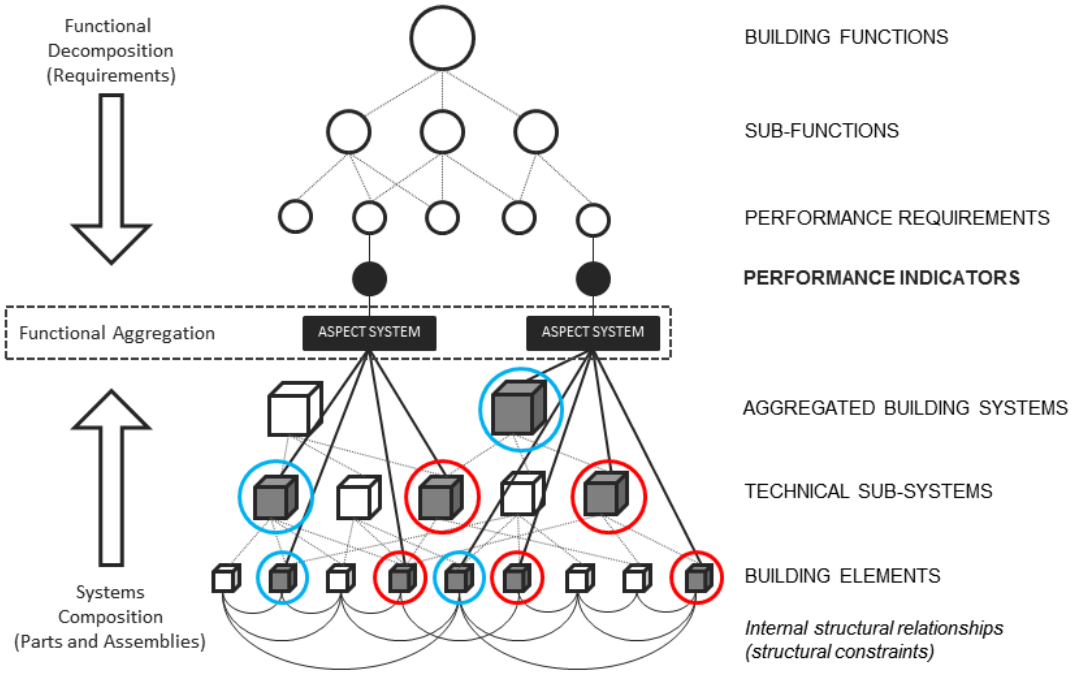

Figure 1.3: Elucidation of aspect systems based on the mapping between decomposed functional requirements and aggregated building systems. Modified from Augenbroe [16].

[1] The relationship between functional requirement and performance requirement would be discussed at the end of Chapter III - Function in the Industry Foundation Classes (IFC)

Notice that querying about the inverse relationship should also be supported. That is, to return the set of functions a given building element participates in. This can also be described as the set of functional roles played by a building element.

- Given a building element a', part of a design model a, return the set of functions in which a' participates.

From this, more specialized queries can be formulated by restricting it to either positive or negative participation, as well as by the type and degree of the participation.

Currently, there is no means of formulating such type of query in CAD or BIM applications. As discussed in section 1.1, providing an timely answer for this type of query to designers and other decision-makers would allow to convey the rationale for previous design decisions, in such a way that potentially problematic design changes could be avoided or properly handled. This is especially true in cases where previous designers intentionally choose certain design configuration or systems integration approach, but for which there is no reliable way to convey the intent behind such decisions, or their downstream implications.

1.5 Research methodology

In order to provide computational means to answer the type of queries described above, the development of a representational framework is proposed according to three main high-level parts:

- Model-based, formal representation of functions (required and supplied).
- Functional and behavioral characterization of structural models.
- Encoding of functional and behavioral relationships in OWL-DL

1.5.1 Model-based representation of functions

The first part deals with the representation of the top part of the diagram in figure 1.3. The goal is to support the definition of requirements models that can be processed computationally.

However, it is not in the scope of this research to provide computational support for the process of functional decomposition itself. As it has been discussed by Augenbroe [16], a high-level functional requirement may be decomposed in several different ways. The variability depends on the specific conditions of the problem, the priorities of stakeholders and the experience of designers, among other ad-hoc conditions. This makes the problem of functional decomposition difficult to formalize and automate in a repeatable way. Furthermore, the theoretical limitation of the method of functional decomposition has been analyzed from a mereology point of view by Vermaas [316, 314].

Therefore, the focus of this research is on functional modeling capabilities able to support description of individual, low level functional requirements, independently of how they got transformed from high level goals or functions (i.e. decomposed or otherwise). Whereas a requirement is not atomic, it can always be subject to further refinement or decomposition. However, this should only take place if it is considered relevant to stakeholders, otherwise no further processing is needed. This imposes the need for functional modeling to support reasoning over these intermediate level specifications, for it is at these levels where many behavioral interactions across systems may take place.

Once identified as having a high priority, the formalization of a functional requirement should involve its description in terms of a behavioral model [70]. More specifically, it should involve a model-based representation of the set of *output behaviors* expected from a system under normal situations.

The notion *output behaviors* is based on research in the area of Knowledge-based Artificial Intelligence dealing with the problem of representation of design functionality, and methods to provide automatic reasoning in support of different types of design tasks.

In particular, the conceptualization of function adopted in this dissertation, including the notion of *output behavior*, is based primarily on the Functional Representation scheme (FR) developed by Chandrasekaran et al.[74, 76], and the Structure-Behavior-Function (SBF) scheme developed by the Goel et al. [151, 150, 256, 155].

According to these schemes, an *output behavior* is a behavior caused by the internal structure of a device or system, while working under certain operational conditions, and

which have external effect in outer environment of such device or system. A device or system may exhibit multiple output behaviors. A function then, is considered as subset of these output behaviors, that is needed or desired by an intentional agent. In this dissertation, the term 'intentional agent' will be used interchangeably with the term 'stakeholder' usually adopted in interdisciplinary design settings to indicate intentionality as it pertains to different life-cycle requirements of a product [328, 248, 296].

The intuition is that devices or systems cause many effects during their use or operation, few of which are intended, while the rest can be considered side-effects. Intended behaviors / effects are also said to play a functional role in the satisfaction of the need or desire [200, 199]. The problem however is that classification between intended versus unintended effects (i.e. side-effects) is not trivial, given that different stakeholders, with different viewpoints and needs, may intend conflicting behavioral subsets. Functional modeling in Building Design therefore should support specification of functional requirements according to these stakeholders' specific viewpoints, so that identification of interactions and potential conflicts could be facilitated through automated reasoning methods.

1.5.2 Functional and behavioral characterization of BIM models

The research proposes the development of semantically richer BIM models, by adding a common formal description of functional and behavioral models associated to both functional requirements and building elements functionality. The extended description follows the FR scheme and its formalization according to the DOLCE ontological framework developed by Borgo et al. [27]. A review of the Functional Representation schema (FR) is introduced in 4.2, while its formalization under the DOLCE upper ontology will be presented in Chapter 5, 5, subsection 4.2.2.

The semantic extension proposed here is intended to be loosely integrated, in order to support the composition of functional and behavioral models abstractly and independently from requirements and structural models (i.e. design BIM models). The goal is to allow these functional and behavioral models to be customized and dynamically linked to both requirements and design building models in an as-needed basis.

Automatic inference of behavioral interactions in turn is provided by reasoning capabilities of the underlying OWL-DL ontology model (see section 1.5.3). The inference process is based on a series of basic assertions provided in both the requirements model (described in the previous subsection), and the design model. Some assertions are predefined at the level of element types, and carried over subtypes by inheritance (e.g. from general walls to load-bearing walls and partition walls), while others are asserted by designers (e.g. vibration insulation walls). The inference on functional and behavioral properties of design models are based on three abstractions defined as fundamental ontological relations:

1. Generalization / specialization (*is-a* relation)
2. Composition / decomposition (*part-of* relation)
3. Participation / participant (*participates-in* relation)

Where *participation* is a relation that holds specifically between an object and a process, action or event. In DOLCE, the former is a type of entity defined under the general category of endurants (i.e. ENDURANT), while the later are entities that belong to the general category of perdurants (i.e. PERDURANT).

This distinction is crucial for the formulation of this research. In particular, the hypothesis is, following DOLCE, that a functional requirement is always the specification of a class of perdurants (i.e. event, process or phenomena etc.) that is intended by a stakeholder for the achievement of her high level goals. Therefore, the participation relationship between structural elements of design (i.e. endurants) and required functions (i.e. perdurants) is not necessarily predefined or fixed. Furthermore, in complex systems such as buildings, the cardinality of functional relationships is rarely one-to-one, but often one-to-many and even many-to-many.

This observation provides the a starting point for unintended behavioral interactions and side-effects to be accounted for in a computational model of design. At the implementation level, main causal relations, as well as general patterns of interactions can be defined abstractly, and further specialized as needed by means of inheritance.

In summary, the proposed approach provides three modeling advantages over an deterministic, one-to-one ''*satisfaction*" relationships:

- A required (functional) perdurant may have more than one participant element (endurant).
- Both intended and unintended participation can be described, either by assertions or by inference.
- Both negative and positive participation can be described, either by assertion or by inference.

Additionally, a fourth auxiliary relationship is proposed in this research for the inference of partial-orders that hold among perdurants. This auxiliary relationship is aimed to provide the explanatory component of the inference process:

1. Predecessor / successor (*hasPrecondition* relation)

The theoretical basis of the distinction between endurants and perdurants, as well the type of ontological relationships that hold between them will be provided in more detail in section 5.1.4, of the preliminary hypothesis, in chapter 5. For the moment, it suffices to illustrate the differences between the categories and relationships being represented in current CAD and BIM applications, and the proposed representational framework (Figure 1.4). The difference stems from different ontological commitments of each underlying schemes [286].

1.5.3 OWL-DL inference

The research proposes the use of Description Logics (DLs) for the machine-readable representation of functions and behaviors associated to design elements. While the particular focus is to support interdisciplinary design of buildings, it is anticipated that the proposed framework based on DL could be generalized to other forms of design.

Description Logics (DLs) are a family of knowledge representation languages commonly used for ontology modeling, knowledge bases and semantic web applications. Most of DLs

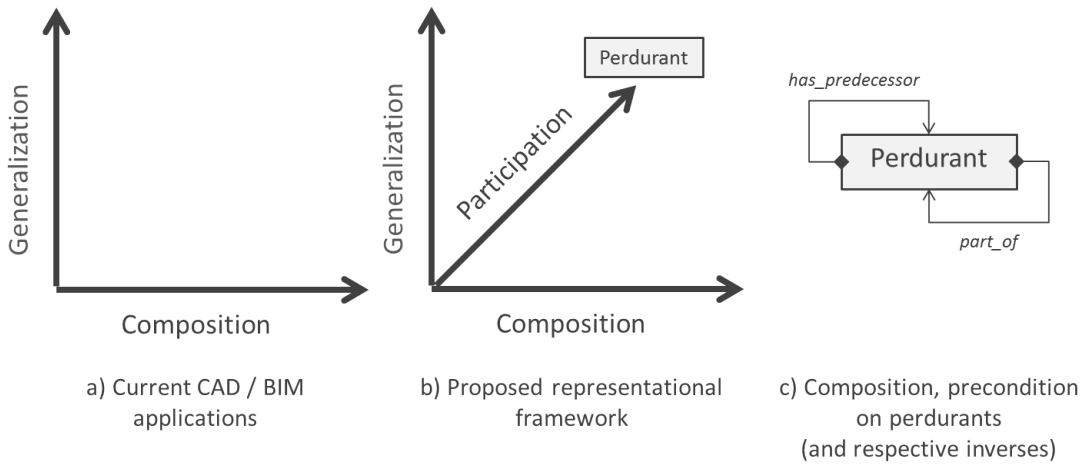

Figure 1.4: a) Ontological commitment of current CAD / BIM representational schemes. b) Ontological commitment of proposed framework that includes perdurants. c) Relationships that hold between perdurants.

are decidable fragments of First-order Logic (FOL), which are characterized with a formal semantics. A properly formulated DL ontology has a very precise meaning that is both human and machine-readable. That meaning is conveyed by a series of DL axioms, which in conjunction specify the logical consequences of an ontology. By running specific types of algorithms called reasoners, additional information can be logically deducted from the set of facts asserted in the model. This capability of producing automatic inferences is the key feature that distinguishes DLs from other modeling languages such as UML [210].

Another important characteristic that makes DLs particularly useful is that DL models are based on the so called Open World Assumption (OWA) principle. This means that DLs provide the flexibility to represent and operate with incomplete information. This in turn makes the functionality of DLs ontologies very different than traditional relational database schemes, which are based on the principle of Closed World Assumption (CWA). In practice the difference lies on how a missing piece of information is dealt with. In the OWA, the missing piece of information will be treated as unknown fact rather than false information. A system built on a CWA will treat the missing piece of information as false, thus rendering the state of the database as invalid [179].

Because of the Open World Assumption, useful logical inferences can still be made, even

with incomplete information. Because of this capability, DLs have been used extensively for Semantic Web applications. The same capability to operate and reason with incomplete information is what makes DLs potentially useful in the context of a design process, especially when much of the information is still in flux.

In 2009, the World Wide Web Consortium (W3C) released the second version of the Web Ontology Language (OWL 2), which provides a Description Logic version called OWL-DL [209, 210]. Since then, a series of OWL based DL reasoners have been developed. Because of the availability of several OWL open-source applications, this research adopted OWL-DL as main knowledge-representation language.

1.6 Scope and expected contribution

The research proposes the development of semantically richer CAD / BIM models, by adding a formal, machine-readable characterization of various functional and behavioral aspects of building systems and components. This additional level of semantics is intended to support interdisciplinary collaboration in building design, particularly in relation to problems of systems integration under multiple criteria of performance. Key to this process is the identification and management of behavioral interactions that occur among different building systems as the design process evolves. This activity is considered important in the context of performance based design, where the satisfaction of a series of different, and sometimes conflicting life-cycle performance requirements needs to be addressed collaboratively by careful functional integration of different building systems.

Altogether, semantically enhanced BIM applications with addition of functional modeling and reasoning capabilities can contribute to the development of various design activities, including:

- Model-based, machine-readable specification of functional requirements.
- Computational support for dynamic identification of input sets required for design-analysis integration.
- Incremental identification of emerging behavioral interactions and functional conflicts

impacting satisfaction of performance criteria.

- Explanation and justification of previous design decisions (i.e. design rationale) to support conflict resolution based multi-criteria trade-off analysis and decision-making.
- Validation of model consistency according to functional semantics.

The main challenge associated with a formal, operational robust functional modeling framework for building design has to deal with need to support multiple functional viewpoints, at different levels of abstraction. Furthermore, this support has to be flexible enough to accommodate the incremental, ill-define nature of building design processes and workflows. For that purpose, the research adopts the general meaning of *function as a role* as functional abstraction that could reconcile the various meanings associated with different functional viewpoints. This concept, which will be discussed in more detail later, defines the scope of the contribution expected for this research, which can be summarized as follows:

- **Supply and demand:** To propose a common representation of functions that serve both the specification of required and supplied functions (i.e. provided).
- **Multiple levels of abstraction:** To propose Domain-specific functional viewpoints (horizontal integration) and general functional viewpoints (vertical integration).
  - **Horizontal integration (domain-specific functional meaning):** To propose representations capable of supporting the integration of domain-specific meaning of functions associated to different technical sub-systems at the same level of aggregation.
  - **Vertical integration (hard and soft functions):** To propose representations capable of supporting the integration of functional meaning across different levels of abstraction. This involves the problem of how the semantic content of low-level functions, usually technical and quantifiable (e.g. heat pumps), relate with the semantics of intermediate and high-level functions, which usually are not technical or easily quantifiable (e.g. productivity).

- **Distal functional dependencies:** Two additional functional perspectives required have to deal with how entities that are distant from each other in space or time are described as functionally related.

- **Nominal and anonymous functional roles:** This class of functional viewpoints relate to the fact that many parts of buildings perform more than one main nominal function. At the same time, many functions are affected by the participation of multiple parts, either negatively or positively. The set of these parts is the aspect system of the function, and the specific goal of this research is precisely to enable the identification of these parts.

1.7 Dissertation outline

This research draws from many areas of study concerned with the problem of representing and reason about design functionality in computational terms. The literature reviewed includes a range of computer models and theories developed in architectural and engineering design, systems engineering, artificial intelligence and applied ontology. This allowed to situate the potential contribution of this research within both a historical and interdisciplinary perspectives. It also provided a theoretical foundation to frame the specific research problem within a larger research context. Specific principles, conceptual models and methods provided the foundation for the formulation of the research hypothesis and methodology adopted. The outline of which is organized according to the following chapters:

Chapter II provides a brief overview in the history of building models, in relation to different approaches developed for the representation of functions and functional requirements. Some common trends and patterns are identified and discussed at the end of the chapter.

Chapter III focuses on the Industry Foundation Classes (IFC) as the most important building data model and interoperability standard available for the representation of building models. The analysis focuses on the how different functional and behavioral aspects are handled throughout the IFC model, including the association with external classification systems such as OmniClass^{TM}. Research efforts and commercial applications related to the

specification and verification of functional requirements will be discussed, and alternatives for their improvement will be proposed.

Chapter IV reviews research in the area of Artificial Intelligence, from early teleological models until more recent efforts in the development of computational models of function and behavior in the context of engineering design. In particular, the chapter analyses the Functional Representation (FR) schema and the Structure-Behavior-Function (SBF) schema, which provide the main theoretical reference for this research. Common aspects in the evolution of different models are identified, along with fundamental principles and challenges.

Chapter V introduces the hypothesis and methodology of the research. Prior the hypothesis however, a preliminary hypothesis is introduced, grounded on the FR schema reviewed in the previous chapter. Additional theoretical background is provided, including an overview the of DOLCE ontology. Then, the formalization of FR under DOLCE is reviewed (DOLCE-FR), with a focus on the most relevant axioms selected for the implementation of a proof-of-concept functional modeling framework (Aspecto-FR). With this background, the final hypothesis is presented, along with the description of the research methodology.

Chapter VI introduces a formalization of the meaning of the concept of'aspect system' based on DOLCE-FR. This definition, along with a subset of DOLCE-FR axioms, and additional axioms are translated from First-Order Logic into OWL-DL. The chapter also presents an overview of an envisioned functional modeling framework, followed by a description of the proof-of-concept implementation in OWL-DL, developed for testing and validation of the hypothesis.

Chapter VII presents in detail three different case studies, each composed of a series of design alternatives and query scenarios, developed for testing of the proof-of-concept implementation and validation of the research hypothesis. In particular, the scenarios allow to test the specification of different functional views, and the process of elucidation of aspect systems based on the inference of functional interactions across multiple levels of abstraction.

Chapter VIII provides a final discussion, presented with the addition of a last case study. This is introduced as minimal working example to frame the analysis of the main theoretical aspects addressed, as well as the main limitations identified. The chapter ends with a summary of the general motivations, specific research problem, scope, hypothesis and expected contribution. Finally main limitations and recommendations for future work are outlined, followed by some final remarks.

CHAPTER II

FUNCTION IN EARLY BUILDING MODELS

Commonsense suggests that all artifacts are made to fulfill some type of purpose. From this belief, it follows that a classification scheme for well defined purposes should be theoretically possible. From a computational perspective, such classification would be useful to provide a more explicit and comprehensive description of design, in terms of representing not only the geometric attributes of an artifact, but also its functionality. A critical part in the formal description of functional intent is the provision of references to the main behavioral mechanisms by which the design is supposed to achieve its goals [278, 151, 308]. In this way, computers could 'understand' the design intent from a teleological perspective, and by consequence provide more intelligent support in various types of design activities [71].

The search for a formal, machine-readable description of design teleology started in the field of Artificial Intelligence by the end of 1970's and early 1980's. These efforts focused initially on the formalization of commonsense knowledge and the development of systems capable of qualitative reasoning about physical phenomena (i.e. naive physics) [171, 89, 213, 90]. Since this was an approach that explicitly addressed the cognitive aspects of design, it was seen as a natural fit for the development of software applications to support engineering design by means of computational functional reasoning.

In the AEC domain, parallel efforts also took place for a more formal and comprehensive representation of design semantics, including the description of functionality for building systems and components. However, the focus here was less on functional reasoning and more on product data interoperability. The latter was considered a fundamental prerequisite to enable more effective interdisciplinary collaboration in the AEC industry, by supporting seamless data integration among different applications used in the design, production and operation of buildings [112]. In this regard, a machine-readable representation of functionality offers potential advantages for various business models and work flow scenarios

[147, 110, 267].

Among these, the integration between design and performance analysis activities has special relevance [18]. In this case, the specification of design functionality plays a major role in the allocation of analytical resources required to support decision-making. This includes the domain experts, along with specialized tools, methods and protocols for interdisciplinary collaboration [16]. Ultimately, the specification of performance requirements dictates, along with the budget available for a project, the scope of modeling efforts and data exchange needed.

However, despite important advancements in reasoning and interoperability, the goal of providing a common, formal and machine-readable representation of design functionality has not been reached at the practical level. One of the main problems is that there is no consensus yet, even within individual design disciplines, on a common, general meaning of function [308, 29, 316, 114]. After proceeding with a literature review in the subject, as developed in the fields of Building Information Modeling, Artificial Intelligence, Systems Engineering and Applied Ontology, this dissertation identified three fundamental inter-related reasons of why the representation of design functionality has been particularly difficult:

1. **The impossibility of a single, universal classification schema of design functions.** This is because design functions are not an intrinsic attributes of artifacts, but rather qualities assigned to them by some intentional agent or community [168]. Generally speaking, the attribution is made based on a subset of all the behavioral capabilities of the artifact when placed under certain context of use. It is in the realization of such subset of behaviors that specific needs are *believed* to be satisfied. In this regard, the notion of function shares important similarities to the notion of *'affordance'* proposed by Gibson [146], and initially discussed in the context of design representations by Brown [32]. Given the highly contextual nature of the attribution, there is not always a fixed, consistent mapping between an artifact and its function. The perception that some mappings seem to be more evident than others within certain contexts of use is rather a matter of convenience and custom. Despite of, or precisely because of the putative nature of functions, it is often the case that an artifact may

stand for more than one function during the course of its lifetime, depending on the benefits or *affordances* it provides to particular stakeholders given the circumstances.

2. **Complex functional needs are never satisfied by a single artifact.** Instead, they require the integration with other artifacts and elements of an entire operational environment. This require an aggregated perspective of the overall system, which Chandrasekaran calls *environment-centric* functional perspective. This type of perspective is the basis for interdisciplinary collaboration in the design of complex system such as buildings and similar civil structures.

   In these cases, there is no one-to-one mapping between a function and single component of the system [76]. The reason is that the satisfaction of a functional requirement depends not only on the occurrence of a number of useful *intended* behaviors, but also on the control over a number of possible harmful *unintended* behaviors that may emerge from negative interactions with other parts of the larger system. This often creates a very dynamic network of many-to-many inter-dependencies among different functions and structural components that is difficult to describe in an explicit, prima fascie manner.

3. **Explicit, many-to-many mappings between functions and structures are never complete and stable.** As argued in Chapter 1, the co-evolution of the design solutions (i.e. the solution space) and design requirements (i.e. the problem space) makes the allocation of functional relations subject to constant change during the design process. Thus, any premature attempt of functional fixation for a design alternative turns to be incomplete and brittle. While such fixation often becomes necessary for practical purposes, the ill-defined and ever-evolving nature of design problems demand a more flexible and dynamic approach. In particular, a flexible and incremental mapping between functions and structures opens up the possibility for solutions that are uniquely fit to certain contexts of use [280, 160, 95, 88]. Unfortunately, most attempts of functional characterization in CAD environments rely on rigid classification mechanisms, which provide very limited help to designers.

The general argument of this dissertation is that design environments should provide more flexible mechanisms for dynamic traceability of evolving functional relations. The critical aspect to enable such capability is the provision of *automatic inference* over structural relations asserted in building models. The semantic 'glue' that would allow inference of functional interactions among subsystems is the model-based characterization of behaviors for both structural elements and design requirements using a common, formal ontological framework.

In this regard it is important to notice that most of the attempts of functional representation in Building Design have mostly addressed the behavioral characterization of physical and spatial elements (i.e. the supply side), with little attention on the requirements side (i.e. the demand side). In fact, the efforts identified by this research on the definition of functional requirements actually followed the opposite direction. That is, the definition of functional requirements in structural terms, i.e. structural constraints.

One of the reasons behind this approach seems to be that the translation of functional requirements into structural constraints facilitates parametric verification. For example, functional requirements associated with occupancy of building rooms and other space uses are typically translated into areas, linear dimensions or ratios of some sort. This allows the implementation of rule-checking procedures to perform computation over instance values [98, 332]. Unfortunately, while this approach is certainly useful to automate the verification of certain structural conditions, it cannot ensure the true satisfaction of high-level functional needs, insofar such needs depend on highly contextual and time-dependent phenomena.

Thus, the compliance to structural constraints cannot not guarantee by itself, that visual, thermal or acoustic comfort goals are going to be achieved, nor that compliant geometry will improve productivity or safety of its occupants. At most they can influence the likelihood of such outcomes, as observed from empirical studies and guidelines. Yet, much uncertainty is left, by not acknowledging behavioral interactions and their unique impact on performance outcomes. Furthermore, from a design perspective, the use of structural constraints as specification for functional requirements reduces arbitrarily the problem-solution space, that otherwise could lead to more innovative solutions [70].

If higher levels of computational support in Building Design are expected, then a more rigorous analysis on the ontological status of function is necessary. This challenge is certainly not exclusive of Architecture or Building Design, but shared with most design disciplines. Therefore much could be gained by studying common aspects across design fields with the hope of approaching a general theory of function in design.

The following section presents a review of some of the most important research efforts in the area of Building Design, and the evolution towards comprehensive model-based representations. Many of these efforts eventually led to the development of current BIM technologies, which inherits many of the principles, strengths and limitations discussed here regarding the representation of functionality. Insights, lessons and recommendations will be discussed at the end of the chapter.

2.1 Early integrated design environments

Early efforts on integrated design environments for the building industry started in the 1970s, soon after the seminal development of the Sketchpad system by Sutherland and Coons [190, 84]. Most of these efforts were essentially research-driven, following a very different path than other CAD implementations of that time, mostly led by the automotive and aerospace industry. In the latter, the focus was predominantly on providing advanced geometric modeling capabilities as primary means to support for engineering design [192]. The technologies developed for the building industry instead had a predominant focus on the concept of a central, common design representation, as means to support the extensible development of specialized domain applications [112]. At the core of this approach was the notion that design integration should be enabled by a shared, standard classification scheme.

The following subsections provide an overview on the evolution of the research and development efforts in building product models and integrated design environments. The discussion focuses predominantly on the representation of function, including the representation of functional requirements, as necessary background for the development of the dissertation.

2.1.1 SSHA and OXSYS CAD Systems

The evolution of integrated design environments started simultaneously in United States and in the United Kingdom as result of research carried in universities and government agencies, sometimes in partnership with private institutions and professional architecture offices. Among the most significant British efforts were SSHA and OXYSIS CAD systems [253, 192].

SSH was developed by the architect Aart Bilj and associates at the Architectural Research Unit, at University of Edinburgh in the late 1960s. This project was sponsored by the Scottish Special Housing Authority with the goal of streamlining the design process of housing units, as well as site planning associated to housing real state. Each of these aspects was supported by separate, specialized computer applications. The first application focused on facilitating floor-plan layout, by providing a library of predefined sets of building components, such as walls, windows, doors, cabinets and stairs. The representation of these components was capable of carrying additional non-graphic information. In this way, a number of simple analysis routines could be executed. These included basic calculation of space areas and volumes, as well as more functional oriented calculations dealing with structural loads, heat transfer and sunlight exposure [112].

The OXSYS CAD system in turn was developed in the early 1970s to facilitate the design of hospitals based on a prefabricated construction system called OXSYS, which was developed by the Oxford Regional Health Authority. This construction system relied on a highly modular post-and-beam design scheme on which walls, concrete slabs and a variety of other building components could be attached to as part of an integrated kit. Because of its focus on modular prefabrication, the OXYS CAD system was a natural fit for a database approach, in which libraries of building components could be created and extended for future reusability. It enabled the definition of building entities at an abstract level, from which multiple graphical and non-graphical views could be generated. These views were intended to feed into several applications for different design activities. Some of the capabilities included functional allocation for spaces, automatic selection of components from the database, basic parametric editing capabilities, design rule checking for assemblies,

integrated structural analysis and member sizing, as well as full product specification and documentation for construction.

Other British systems worth of mentioning were HARNESS and CEDAR systems, which also focused on hospital design, based a suite of specialized applications within an integrated framework [253].

2.1.1.1 Functional Aspects in OXSYS CAD

Besides the database of building components, OXSYS pioneered the implementation of a database of project requirements (i.e. project briefs). In this way, simple design properties could be verified in relation to target values defined in requirements database. The main theoretical contribution of the OXSYS CAD system is that it anticipated the adoption of an approach very similar to object-based modeling, being probably the first platform with an integrated suite of applications covering different workflow activities [112].

2.1.2 Space planners and layout generators

In the United States, a strong focus was on the development of systems to support space planning and automatic floor layout generation. An early example of that period was the General Space Planner (GSP) developed by Eastman and his research group at Carnegie Mellon [107, 108]. This system was strongly influenced by cognitive models of how architects solve design problems, along with the lines of problem-solving methods developed by A.Newell and H. Simon, also at Carnegie Mellon University [132]. Other similar automated space planning and layout generators of that time worth of mention are the Design Problem Solver (DPS) by Pfefferkorn [252] and the IMAGE system by Johnson, Weinzapfel and others at MIT [324].

Additionally, some early commercial design systems also started to appear in professional practice. In 1973, the architects at Perry Dean and Stewart in Boston pioneered the adoption of architecture design software developed by Design Systems [192]. The software comprised a family of related applications backed by a centralized database called Compugraph, which could include information about geometric properties and cost per square foot of building spaces. Another application, called Compurelate could automate the generation

of floor-plan layouts in the form of bubble diagrams and simplified rectangular geometries based on adjacency matrices. These matrices were used by the architects to represent the functional compatibility among different spaces. This approach was found useful in the development of large institutional projects with complex programmatic requirements, such as schools and hospitals. According to Stewart [294], the architects appreciated the potential of the system not only to allow the description design goals and requirements in a more explicit form, but also that such descriptions could be easily retrieved to convey the design rationale of past design decisions and for future reuse.

2.1.2.1 Functional Aspects in Space Planners and Layout Generators

In general, all these systems shared a similar approach in relation to the representation of design requirements. Typically, they addressed the description of requirements in terms of structural constraints, which could be more easily resolved by heuristic procedures. Such constraints included rules for spatial location, sizing and adjacency, proximity and orientation, etc. Unfortunately, these systems lacked good user interfaces and geometric modeling capabilities. Furthermore, they lacked a general, high-level definition of a data model. At some point it became apparent that such definition was necessary to support a wider range of integration among different design applications [112].

The realization that a standard, formal data model was needed led to the development of the Building Description System (BDS) in 1974. This was one of the first tridimensional solid modelers with building-specific data representation [192]. Additionally, BDS was a precursor in the development of spatial set operators, as well as in the use of mechanisms for multiple instantiation. However, interactivity was limited by a text-based user interface and a restricted set of geometric operators that could not describe building assemblies with enough level of detail [132].

2.1.3 GLIDE

The lessons learned with BDS were applied later in 1977 for the development of the Graphical Language for Interactive Design, GLIDE by Eastman and his group at Carnegie Mellon University. GLIDE is especially relevant in the history of CAD because it was probably the

first effort to develop a computer language specifically tailored to support design. Thus, GLIDE provided a high-level, fully integrated computing environment for the design of different types of physical systems. It was based on a common set of database features and operations tied to a general-purpose likelihood solid modeler.

Among the many innovations of GLIDE, it featured for the first time parametric modeling capabilities for the interactive modification of tridimensional shapes. In this way, GLIDE allowed the construction of geometric arrangements necessary for the representation of assemblies at various levels of detail, while also supporting derivation of two dimensional projections for drawing documentation [105].

2.1.3.1 Functional Aspects in GLIDE

Another important conceptual innovation of GLIDE that is relevant for this research was the adoption of object-orientation in the definition of design entities. In particular, GLIDE was one of the first systems to refer to geometry as an *attribute* of design objects, rather than a first-class entity in itself. This novel representational approach provided a new level of semantic flexibility, in such a way that design elements could be displayed according to the purpose at hand. Moreover, the semantics of objects could be extended with additional attributes and relationships. This included, in the case of GLIDE, the definition of functional attributes intended to support integration with various types of performance analysis [97].

2.1.4 GLIDE-II

GLIDE-II followed GLIDE in 1979, to satisfy the requirements for a more comprehensive building data model for the U.S. Army Corps of Engineers. GLIDE-II was essentially a programmable database back-end, intended to support the development of various modular front-end design applications. An integrated design environment on top of GLIDE-II was developed by a different research group at the University of Michigan. This system, called Computer-Aided Engineering and Architectural Design System (CAEADS) supported a number of front-end design applications, such as habitability studies, energy consumption analysis and verification of building specification, among others [192].

2.1.4.1 Functional Aspects in GLIDE-II

The main conceptual underpinning of GLIDE-II was the recognition that a fixed, monolithic data model was not practical in Building Design. Instead, the increasing diversity of building technologies, materials and processes with evolving sets of attributes required an entirely different representational approach. By taking advantage of the new ideas from object-orientation, GLIDE-II proposed the representation of multiple, extensible structural hierarchies, with enough flexibility for accommodate the description of different attribute sets and aggregation strategies. Such approach, called "abstraction hierarchies" was deemed especially necessary to deal with the description of various performance aspects associated to Building Design.

For that purpose, GLIDE-II introduced a special type of polymorphism called *type unions*, with the goal of supporting semantic extensibility of models. In this way, elements such as walls or any other high-level data object could have new performance properties appended to it on an as-needed basis. Additional semantically extensibility was envisioned through the ability to establish new relationship types among objects, according to the goals of each project and use case scenario [112]. However, this proved to be a difficult theoretical as well as implementation challenge, and remains to a large extent, an unresolved research issue.

2.1.5 SEED

A more recent attempt to develop an integrated design environment is provided by the SEED project. This was a large multidisciplinary research effort led by researchers at Carnegie Mellon University in partnership with the University of Adelaide in Australia. SEED stands for Software Environment to Support Early Phases of Building Design. As the name indicates, SEED focused on early stages of design, with an emphasis on rapid exploration of design alternatives. A key component for enabling such capability was the use of Case-based Reasoning (CBR) techniques for automatic retrieval and adaptation of design precedents [133, 131].

For this reason, SEED relied on a dedicated module for the representation of design

requirements in the form of an Architectural Program (AP). This module, called SEED-Pro, was critical to provide an explicit description of the design problem in such a way that a series of useful computations could take place. Among these, it was the automatic generation of floor plan layouts and tridimensional configurations. These tasks would be performed in two additional modules, namely SEED-Layout and SEED-Config respectively. The evaluation of performance in turn would be done by external applications. However, an explicit and persistent representation of requirements was thought to facilitate the early identification of conflicts among competing objectives. The resolution however was let to stakeholders involved, given the cognitive complexity of this type of task [2].

In general, SEED followed a similar object-orientation approach that GARM and EDM, to be discussed in the next section. For instance, functional requirements are modeled independently in a separate environment (i.e. SEED-Pro), and then allocated to design entities for generation and evaluation of alternatives. In particular, functional requirements are derived initial specifications and then grouped within objects called Functional Units (FU). Functional Units are represented as a series of specialized objects with constraints placed on attribute values. As usual, a FU can be hierarchically decomposed into lower level FUs. During the allocation to Design Units (DU), constraints specified by a FU are verified against the property values of design elements, including location, dimensions, or other behavioral properties such as load-bearing capability, thermal resistance, fire resistance ratings, etc.

2.1.5.1 Functional Aspects in SEED

The approach of representing requirements as constraints over property values has an enduring tradition in design. The main reason seems to be, as discussed earlier, that such approach facilitates the execution of various low-level tasks, including parametric change propagation and compliance verification (e.g. code checking), insofar these can be reduced to simple quantities or logical values.

In the case of SEED, this approach also facilitate the formalization of similarity metrics for storage and retrieval of precedents within its Case-Based Reasoning framework. thus

introducing a new method for knowledge management within design organizations.

Two other systems were developed later based on the SEED experience, namely, RaB-BiT [119] and DesignTrack [245, 246]. The goal of these systems however was narrower, focusing instead on supporting requirements engineering and knowledge management activities within collaborative settings. Similarly to SEED though, the concept of requirements was conceptualized as quantitative or logical value constraints without any reference to underlying principles of causality.

Unfortunately, while this approach has some practical advantages, it cannot support traceability of functional inter-dependencies. In this way, the effectiveness of these systems to help engineering and management of requirements is questionable, especially in relation to the satisfaction of high-level, inter-dependent functions. Indeed, these require the understanding of causal relationships within specific modes of deployment that cannot be expressed by property values alone. For instance, the single specification of thermal resistance values for external walls cannot, by itself, ensure that the required levels of thermal comfort will be met. This depends, among many other things, on the people and equipment occupying the space, and the activities taking place.

Certainly, part of the challenge stems from the difficulty to specify functional requirements in an unambiguous and operational form. Indeed, Akin at al. [2] recognized that the process of requirements specification is complex because of the open-ended nature of the input data, and the lack of formal methods for mapping from behavioral to functional requirements.

2.2 Building product models

The identification of different design requirements and their many-to-many mappings onto technical solutions involves a significant amount of knowledge integration. A key aspect for the success of the integration is the way in which different types of knowledge are represented during the design process. In particular, one of the main goals of a design representation should be to facilitate the characterization of the dynamic interactions that continually evolve among requirements and technical solutions. This is a fundamental problem from the

perspective of design innovation because new structural compositions need to be explored constantly in order to achieve new types of functionality and levels of performance [110]. This scenario in turn leads to an increasing demand for early and continuous assessment of the impact that different alternatives may cause from multiple perspectives and at multiple scales of a design problem.

These conditions led to the realization that it is very difficult to anticipate all possible types of expertise and domain-specific applications that need to be brought into a given design process. Therefore it became clear that integrated design environments should not be based on closed, monolithic systems, but above all open, flexible and extensible. The focus then shifted towards the development of generic building models that could be integrated primarily to external applications. The consolidation of object-oriented programming languages, relational databases and other technologies paved the way to this new approach.

2.2.1 Engineering Data Model (EDM)

The Engineering Data Model (EDM), developed by Eastman et al. in the early 1990's was an attempt to address these issues, based on several ideas originally explored in GLIDE and GLIDE-II. One of these ideas was that behavioral interactions affecting building performance could only be inferred if knowledge about design intent and functionality were explicitly represented within a shared knowledge-base. Because of the highly contextual nature of both concepts, EDM proposed a modular architecture to support the semantic extensibility and customization of object types, properties and relationships described in the knowledge-base [113].

Within EDM, the specification of functional entities (FE) became the most important unit of design expertise. The goal was that FEs could be composed by experts at multiple levels of abstraction, independently from structural compositions. During design, multiple FEs could be retrieved and dynamically linked to structural elements as the design proceeded. In order to focus only on potentially relevant functional relationships, FRs were specified using sets of constraints called 'accumulations'. An accumulation can be described

as a model view that is relevant from a given functional perspective. As result the accumulation only returns a subset of the entire design model meeting certain preconditions deemed necessary for the verification of performance. For example, the performance verification for a truss joint requires at the minimum, the topological connectivity of its members and the convergence of loads. In EDM these items are specified as part of an accumulation definition (see Figure x). As in any other accumulation, this could be extended and customized in order to meet new design conditions and requirements. [113].

The load conditions of the joint in turn could be affected by other structural elements or configurations being created in a different part of the design model. Because the of the impossibility of anticipation all possible behavioral interactions arising from design changes, EDM pioneered the use of automatic reasoning techniques for the inference of new relationships at the instance level. Such capabilities were based on a First-Order Logic (FOL) axiomatization of EDM's knowledge-base and the use of Prolog for reasoning. Thus, new functional relationships potentially affecting performance could be inferred automatically as result of the combination of axioms contained in the various EDM knowledge modules and instance data created by the designers during the course of a project [103].

2.2.1.1 Functional Aspects in EDM

The idea that functional models composed abstractly and independently from structural models was initially proposed in the area of Knowledge Representation, a sub-field of Artificial Intelligence, and introduced in the representation of buildings by Gero [142]. SEED discussed previously, and GARM, discussed below, were based on similar conceptual lines. However, EDM was different because it was the first system to make use of automatic logic inference to dynamically capture the semi-lattice of functional inter-dependencies that otherwise would remain tacit to designers. The use of reasoning capabilities was seen as a powerful approach to cope with the complexity of the design process, and the often unpredictable conditions under which design innovation and creativity emerge.

This can be seen as an early reaction against the "kit-of-parts" modeling approach predominant to this date in the implementation of various building models. In this view,

design functions are assumed to stand in a fixed, one-to-one relationship with building elements in a very prescriptive fashion. Unfortunately, this not only restricts the scope of design possibilities, but it also increases uncertainty over performance outcomes. By using reductionist assumptions regarding function, a series of behavioral interactions can be easily overlooked, often with negative consequences.

2.2.2 The ISO-STEP model

During the early 1980s' the main approach to support data exchange between product design applications was the implementation of specialized translators. However this was an expensive approach that relied on the development of a series ad-hoc point solutions. As soon as new software were developed, a whole new set of specialized translators had to be implemented with little or no compatibility with prior ones. The brittleness of such approach made clear that product data definition should be standardized, so that different engineering applications could be communicate with each other based on open industry protocols.

This realization led to the development in the US of the Product Data Exchange Standard (PDES) initiative, while that in Europe the International Standards Organization (ISO) started the STandard for the Exchange of Product Model Data (STEP), also known as ISO 10303. Eventually PDES and STEP merged into a single standardization initiative to support data exchange required for product design and manufacturing. According to Eastman, there were several objectives guiding this effort, based mostly on new advancements in computer science, database and software engineering being developed at that time [112]. These objectives were:

- To adopt new concepts from object-oriented programming (e.g. inheritance).
- To develop formal specifications using new data modeling languages.
- To separate the data model from the physical file format.
- To support alternative physical level implementations.
- To enable data model views (i.e. subsets) based on specific information needs.

- To capitalize from reference models that are common across domains.

Particularly influential in the concept of STEP was the layered architecture of relational databases, promoted by the American National Standards Institute (ANSI-SPARC) [112]. This architecture establishes three levels of data abstraction, namely the physical, logical and application layers. For the purpose of this research, it is important to focus on the last two.

The logical layer establishes the semantics of the information to be represented in an implementation-independent way. The application layer is written on top of the logical layer, as mechanism to define subsets of the logical model that are relevant for specific applications (i.e. model views). A fundamental step in the definition of the logical layer is the conceptualization of the domain, also called as *Universe of Discourse* (UoD) [117]. This step essentially means the formalization of expert knowledge about some aspect of product to be represented (i.e. the domain) using well defined modeling methods. For that purpose the STEP committee adopted IDEF1x and NIAM as primary modeling languages, adding EXPRESS-G to the specification later on. For a machine-readable specification of the logical and application layers, EXPRESS was adopted as language of choice.

The representation of different dimensions related to product development, from design to manufacturing and beyond was addressed by STEP in a highly modular manner. Indeed, STEP did not aim to define models for whole products, but rather a set of middle-level definitions, each dealing with a particular facet of the product's life-cycle. In this way different modules, called Application Protocols (AP), could be combined to create data models tailored to specific domain applications. Each AP contained both the conceptualization of an aspect of the product being addressed, using one of the aforementioned data modeling languages, and a machine-readable specification in EXPRESS to guide the implementation of various software applications. The first part of the AP dealing with the conceptual specification is called an *Application Reference Model*, or *ARM*, while the second part is referred to as *Application Interpreted Model*, or *AIM* for short. Altogether, they define the information structure required to characterize the semantics of a product model.

Additionally, the modular architecture of STEP was intended to facilitate revision and

extensibility of APs according to new industry requirements. Following these principles, and after many years of development, a significant number of APs was formulated, covering a wide range of domain applications. These include electronic devices, ship building, aerospace and automotive, oil and gas plants, building construction and furniture, systems engineering and design, among others.

2.2.2.1 Discussion on ISO-STEP

The pressure to demonstrate useful results early on forced the development team to adopt a very pragmatic approach in the formulation of STEP. Thus the effort was guided by modeling paradigms and technologies already available and well-established in the industry, avoiding the need for further theoretical inquiry. The adoption of existing modeling paradigms favored a bottom-up definition of domain specific APs, in the hope that middle-level integrations would eventually happen according to industry needs. Unfortunately, there was no general framework to guide the harmonization and extensibility of model. Moreover, the proliferation of APs often led to duplication of efforts, redundancy of contents and semantic inconsistency [112]. In practice, the middle-level integration of APs was done in a piece-meal fashion, supporting only fix sets of applications with well-defined data exchange scenarios.

In the field of building product modeling, these special-purpose model subsets were called *partial models*, *view type models*, or *aspect* models [25]. A major effort to define industry specific partial models was CIMSteel, which addressed workflows and associated information requirements for the steel industry. Another important partial model was COMBINE, which focused primarily on the integration between design and energy analysis applications.

The underlying assumption behind the idea of aspect models or model views was that data exchange scenarios were more or less the same for certain types of building and design workflows. Hence, the specification of model views could be prescribed by simply conforming with the data requirements of the receiving application. However, this implies a reductive approach, where the design problem itself - as translated into a set of functional requirements

- gets equated to the data requirements of a receiving application. Unfortunately, such assumption is questionable, especially in situations where the design problem is unclear and functional requirements are subject to constant change.

In such cases, it is not possible to totally prescribe application workflows and data exchange requirements with certainty. This is particularly common in the AEC industry, where not only design requirements tend to change, but also business constraints and resources can vary significantly between projects.

These problems made clear the need for a more general conceptual framework, with well-defined criteria for definition and integration of product models with a larger and possibly extensible suite of applications. In the building product modeling community this was addressed by a number of research efforts, both within and outside of STEP [97]. One major example under STEP was the General AEC Reference Model (GARM). Others were conceptually related to STEP by using NIAM and EXPRESS as their modeling languages of choice. These models include ATLAS, the AEC Building Systems Model, the RATAS model from Finland, and the Building Construction Core Model (BCCM) [25, 97, 112].

In the next subsection an overview of the GARM framework will be provided, and its approach towards the representation of functional requirements in the context of the AEC projects. It is followed by an overview of COMBINE. While COMBINE initially focused on the definition of aspect models for energy analysis applications, it expanded later to address the general problem of design-analysis integration from a process-centric perspective. For that purpose it introduced the use of process models to capture the conditions under which different data exchanges could happen. Both GARM and COMBINE proposed the explicit characterization of functional requirements as mechanism to guide the formulation of aspect models.

In the particular case of COMBINE, another important conceptual contribution was the recognition of the intrinsic role that the specification of functional requirements play in the overall structure of collaboration in design.

2.2.3 The General AEC Reference Model (GARM)

The General AEC Reference Model (GARM) was originally proposed by Gielingh [147] to be part of the ISO 10303 (STEP). As introduced earlier, the main goal of STEP was to provide a comprehensive standard for product data representation and exchange to be used by a variety of industries and product types. In this context, the main purpose of GARM was to work as a high-level "integrator" to facilitate the planning and coordination among different domain-specific sub-models developed under the STEP umbrella [17]. To satisfy this need, GARM addressed product development from a life-cycle perspective, identifying a series of life-cycle stages and stage transitions. These were then used to characterized different information requirements and to guide the integration of various STEP resources. Underlying these ideas was the formulation of three general types of abstraction, intended to provide a complete description for any kind of manufactured product. These included:

- Generalization / specialization
- Aggregation / decomposition
- Characterization, which describes characteristics or *aspects* of design that deemed important for some stakeholder.

Within this framework, a GARM model can be described by a generic entity, called Product Definition Unit (PDU). A PDU represents a product or a particular decomposition of the product. Each PDU in the decomposition hierarchy is divided in two main parts. The first part is called a Functional Unit (FU), which provides the description of the functional problem or requirement to be satisfied. A FU also has a reference to the stakeholders and life cycle associated to the requirement. The second part is called the Technical Solution (TS), which is a reference to a candidate system or design component that may potentially fulfill the requirement. A FU may have zero, one or more Technical Solutions. In the first case, the FU requirement is likely to be excessively difficult and need to be relaxed, while that in the latter case, a criteria need to be defined in order to select the best option among several candidates. Each design candidate TS in turn can be described by a set of characteristics,

which refer to different aspects of the product, such as strength, durability, cost, energy consumption, acoustics, etc. The diagrammatic expression of this FU-TS coupling, known as the Hamburger model, is showed in the Figure 2.1.

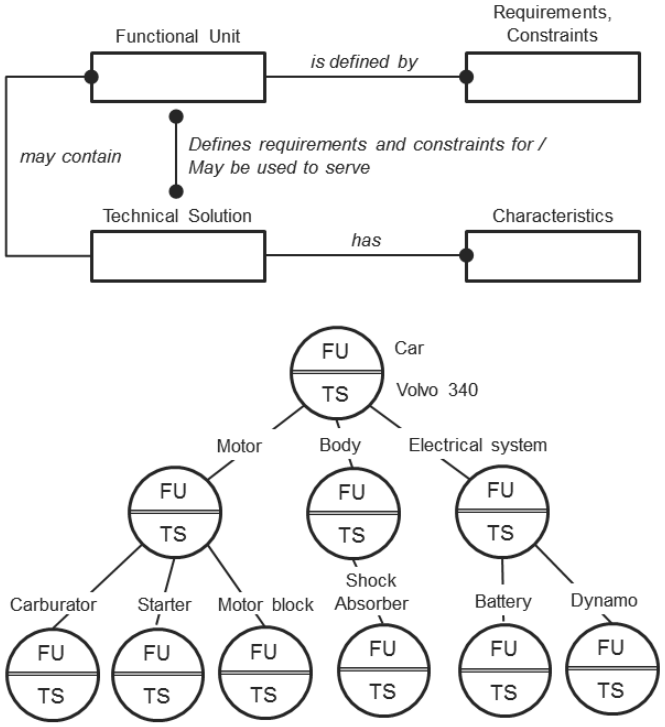

Figure 2.1: General AEC Reference Model (Hamburguer Model). Modified from Gielingh ([147]).

2.2.3.1 Functional Aspects in GARM

The intent behind the Hamburger model was to provide a systematic, rigorous and machine-readable information model to support the design process. To achieve this goal, the design process was assumed to be akin to problem-solving, and therefore susceptible of a 'divide-and-conquer' approach[147].

Unfortunately, the tree-like decomposition originally proposed by GARM and its narrow focus on one-to-one mappings between requirements and solutions seems insufficient to deal with the complexity of real-world design scenarios. In particular, there is no explicit recognition of the semi-lattice structure of functional inter-dependencies among requirements

and technical systems. Moreover, the GARM model does not seem to provide the degree of flexibility required to capture the evolution of such functional inter-dependencies overtime. These limitations hinder the prospects for multi-criterion evaluation of performance, and by consequence, the effectiveness of interdisciplinary collaboration.

2.2.4 COMBINE

The need for more effective multi-criterion evaluation of building performance became evident, as many of the earlier evaluation tools failed to address the role that functional inter-dependencies play in various performance aspects of buildings. In general, those tools resulted mostly from mono-disciplinary R&D efforts, with a narrow focus on very specific types of performance, and little consideration about their usability and integration from a design workflow standpoint. [18, 11]. Indeed, such an approach could only be used reliably in situations where performance requirements were relatively simple and boundary conditions were well-understood from the outset. Normally, these conditions apply in cases of routine or repetitive design, where innovation in building technology is less of a driver.

Where design innovation is an important, a much more careful attention is required over the dynamics of the design process itself. In particular, a better understanding of the evolution of requirements and their set of functional inter-dependencies is needed. The COMBINE project (Computer Models for the Building Industry in Europe) was developed in the early 1990's, with the goal of setting the theoretical and methodological foundations for a process-driven integration framework. To that end, COMBINE focused on the systematic characterization of design stakeholders (i.e. actors) along with the set of performance requirements, tasks and information workflows associated to each stage of a project life-cycle.

COMBINE started as part of JOULE, a larger European research program dedicated to aspects of energy production and efficiency. For this reason, COMBINE focused initially on integration of Building Design with energy analysis applications. This first phase, called COMBINE-1 was eventually succeed by COMBINE-2, which considered a wider scope of performance analysis applications [24].

The emphasis on a process-driven integration of multi-criterion performance evaluation in COMBINE-2 came from the realization that the scope of data modeling is a function of the different views that need to be supported during a design process [17]. Since each building project has a unique set of requirements, resources and constraints, the resulting workflow and modeling efforts are also unique. Thus, a higher level of representational flexibility is required to accommodate different workflow scenarios.

To that end, COMBINE proposed a central conceptual data model, the Integrated Data Model (IDM), from which domain-specific views called Aspect Models, could be extracted on-demand and fed into different evaluation tools. The main criteria for the generation of such views were the type of performance evaluation to be executed and the stage of the design process under which such evaluation was needed. Hence, particular views could be generated from a number of different functional perspectives, and according to increasing levels of granularity.

One of the modeling principles adopted to enable this capability was the separation of functional specifications from the description of geometry, as seen before in SEED, EDM and GARM models. Relationships between these two types of entities could be established explicitly through the use of the 'satisfaction' abstraction. Similarly to the GARM ontology, COMBINE also made use of the 'characterization' relationship in order to derive specialized views from the model.

These specialized views were called on COMBINE 'aspect models', and are somewhat similar to the concept of 'accumulation' used in EDM. Indeed, both concepts refer to a special type of aggregation abstraction that differs from traditional physical composition defined by conventional structural parthood relationship (e.g. physical or spatial assemblies). In both models, an aggregation is a view-dependent abstraction, in the sense that a structural entity, i.e. a technical system, is said to belong to an *aggregate* only if it has a *characteristic* that is relevant for some stakeholder. Since a structural entity may have multiple characteristics (e.g. color, transparency, strength, cost etc.), it may belong to more than one aspect model at the same time.

In principle, it is this one-to-many relation cardinality between a structural entity and

possibly multiple aspect models what constitutes the foundation for the multi-criterion evaluation and decision-making framework proposed by COMBINE. As new structural entities are added to the design, with possibly new sets of characteristics along the way, new behavioral interactions and conflicts may emerge that require a new round of performance assessment and resolution.

As mentioned before, the driving criteria for the generation of aspect models was the type of performance evaluation to be executed at a given time. At the implementation level, an aspect model works as the main source of input data for the execution of analysis applications. Therefore, multiple aspect models with associated analysis applications need to be defined and invoked in order to support the intended multi-criterion framework.

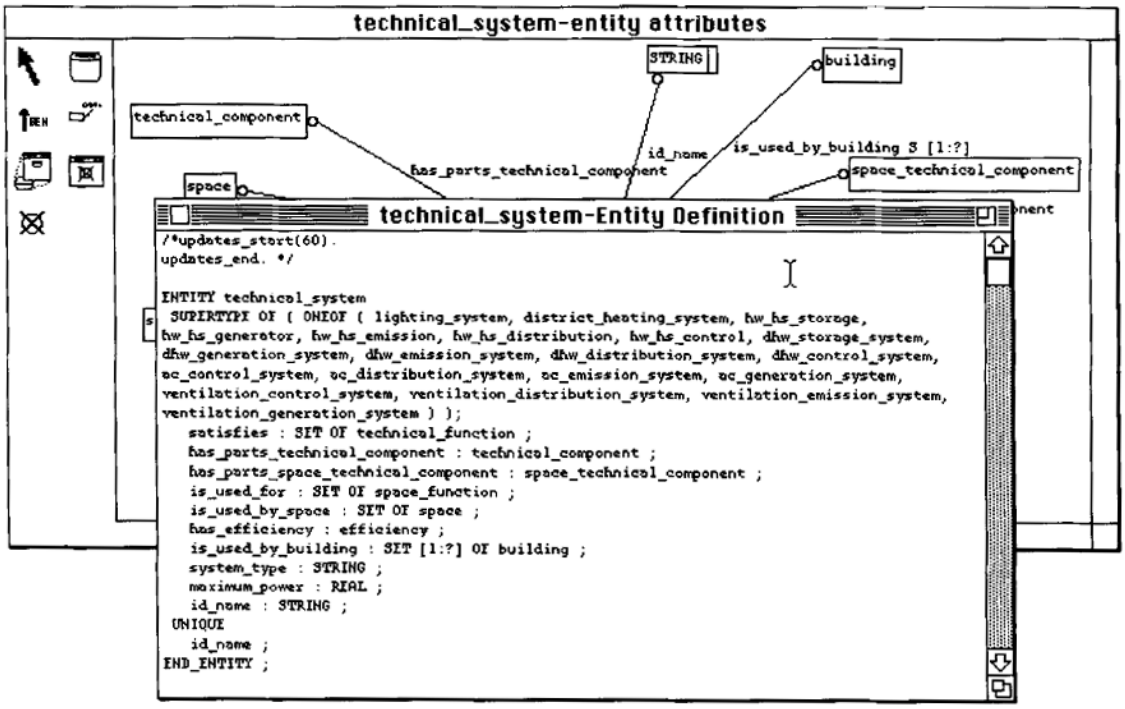

Figure 2.2: Definition of a technical system in COMBINE using STEP. From Amor et al. [11], p. 181.

2.2.4.1 Discussion: Functional Aspects in COMBINE

The theoretical contributions of COMBINE were not only in the development of a product data model to support the integration of design and analysis applications, but also,

and perhaps more importantly, in addressing the role that such integration should play in improving collaboration and decision-making during the entire life-cycle of project. At the core of this relationship is the realization that design requirements are fundamentally multidimensional, with increasing number of behavioral interactions as the design process evolves. This in turn demands an incremental, multi-criterion approach towards the evaluation of performance, so that functional conflicts can be assessed and trade-offs resolved collaboratively in a more systematic manner.

At the conceptual level, COMBINE also relied to some extent on ISO-STEP as conceptual framework. In particular, it partially adopted NIAM, EXPRESS and EXPRESS-G as conceptual modeling languages. This was intended to facilitate a future integration with the larger interoperability effort behind STEP. However, the decision also imposed some important limitations. In particular, it enforced the need for hard-wiring the definition of aspect models at the schema level, thus restricting the semantic scope to few pre-defined types. Moreover, structural entities had to be strictly typed at the instance level in order to be recognized as valid members of an aspect model [11].

This last condition contradicts the goal of a truly flexible representation, where membership criteria of an aspect model should be determined by the functional roles that structural entities play within a specific context of use, rather than by hard-wired typing. While this consideration applies to the entirety of the design process, it is especially relevant during conceptual stages, where the functional meaning of many design elements tend to remain ambiguous and malleable. In this situation, designers are more interested in exploring what elements can do under different circumstances, rather than in what they actually are, i.e. their type.

This kind of semantic dichotomy or ambivalence seems to be critical for the creative process. However it makes the definition of product models, particularly for early performance evaluation, extremely challenging. The adoption of the 'characterization' abstraction in GARM and COMBINE was an attempt to overcome this problem. However, the lack of a more robust ontological definition for what is a 'characteristic' remained a common weakness in both models. In particular, there was no clear distinction between structural

and behavioral characteristics. As result, it is not clear how a member of an aspect model participates in the phenomena associated to a performance outcome. Thus, the whole principle of causality intrinsic in the concept of an aspect model remained computationally undefined.

2.3 Discussion on early building models

Early research in integrated environments for Building Design was strongly influenced by the problem-solving paradigm emerging in the area of Artificial Intelligence. Among the various problem-solving methods developed at the time, constraint satisfaction was especially attractive, because constraints provided a convenient way to represent design requirements in a machine-readable format.

However, the application of the problem-solving paradigm to Building Design, along with the interpretation of requirements as equivalent to constraints presupposed two main tacit assumptions. First, that the problem definition is known from the outset and not subject to change. Second, that constraints and properties of candidate solutions can be expressed in similar terms. In this way, verification of design alternatives could be done by simply checking their property values against corresponding constraints of the problem formulation.

Unfortunately the implication of this approach is that not only the problem space has to be known beforehand, but also the solution space as well. In other words, the interpretation of requirements as constraints is no longer a problem formulation, but rather a premature specification of possible solutions. This view can indeed be found in all design frameworks that follow a 'kit-of-parts' approach. In such cases, the design process involves mostly the composition of predefined sets of elements, each with a very specific functional meaning, and according to a very limited set of compositional rules [103]. By doing this, the representational expressiveness of the design environments get effectively reduced. Only under these deterministic set of conditions that the mapping between problems and solutions can be established procedurally.

Such trading-off between representational expressiveness and procedural efficiency has

been beneficial in many cases, especially during the early days of CAD. Among its advantages, it enabled a tight integration between design and analysis applications. This approach eventually led to the development of 'monolithic' environments, which in theory facilitated the development of additional layers of service and functionality. Some of these efforts included the use of Artificial Intelligence methods for automatic retrieval and adaptation of design precedents under a Case-Based Reasoning (CBR) framework.

Unfortunately, this tightly integrated, monolithic approach ended up being too brittle to accommodate the expanding sets of design requirements, applications and real-world use case scenarios. The complexity associated with business requirements and workflows shifted the interest towards a more general building model framework, able to support more flexible and open-ended integration of different domain applications. For this purpose, data interoperability was a seen as fundamental precondition, motivating the development of various building data models.

The increasing adoption of object orientation (OO) in programming and data modeling provided the conceptual framework for that effort. Abstractions such as inheritance, encapsulation, composition and polymorphism expanded the range of semantic relationships that could be represented more explicitly. This in turn facilitated modular extensibility of product model definitions, including the ability to add new object properties to fulfill specific data requirements.

Systems like GLIDE and GLIDE II pioneered the implementation of some of these ideas very early. In particular, GLIDE II recognized the need for semantic extensibility via addition of functional attributes at different levels of the building system hierarchy. Moreover, GLIDE-II introduced a new aggregation abstraction called "abstraction hierarchy" to provide function-specific views of the design model for performance evaluation. However, this idea required a more formal criteria for classification of building functions and functional relationships that proved too difficult to define a priori.

The EDM model took a step further by developing a more flexible and open-ended mapping between functions and associated model views. To this end EDM relied on functional models that could be modeled independently of structural models. Mappings between the

two types of models were made during the design process using a deductive reasoning engine. Functional aggregations or views, called 'accumulations' in EDM, were derived based on logical inferences about functional dependencies among design elements. Unfortunately, limitations in software and computer power at that time interrupted a line of development which otherwise seemed very promising.

However, the functional criteria behind the definition of model views gradually morphed towards a more pragmatic set of issues. With the introduction of STEP, the need to support data interoperability across various manufacturing industries took a prominent role. In this context, the need for standardization of product models and data formats became the first challenge. Eventually, the semantically rich notion of model views, originally understood in EDM as functionally driven abstractions, started to be seen simply as an useful construct to support specific data exchange scenarios. The reliance on relational databases reinforced such approach, by providing a method for definition of model views in the form of derived tables.

As result, this shift in criteria relegated the representation functional aspects in Building Design to a secondary role. A counter effort to this trend was the COMBINE model, which insisted on the importance of functional criteria as main driver for the definition of model views. While COMBINE acknowledged the role that design workflows and data exchange requirements should play at the implementation level, it also stressed the need for a conceptual framework to guide such implementations. In particular, such framework was seen as necessary to address the multi-dimensional nature of Building Design, where multiple and often conflicting sets of functional requirements demand a higher level of coordination and decision-making. Without such framework, the value of data exchange and interoperability is diminished.

The functionally driven definition of model views, called aspect models in COMBINE, was an attempt to provide the basis for such a framework. The use of the 'characterization' relation was used to set the conditions under which a design entity should be considered part of an aspect model. However, there was no explicit differentiation in COMBINE among different 'characteristic' types, e.g. between structural and behavioral characteristics.

Since the definition of an aspect model is fundamentally teleological (i.e. the aspect system), this lack of differentiation seems problematic. In other words, it is not clear how structural characteristics such as texture or color may impact the satisfaction of a functional requirement, if the causal behaviors intrinsic to those characteristics are not made explicit within a given context of use...

The theoretical challenges associated with this problem remained, to a large extent, under-addressed. Instead, a more pragmatic approach towards the development of general building model framework was adopted. Eventually this effort led to the development of Industry Foundation Classes (IFC) model, with a strong focus on data interoperability. To this end, the definition of new property sets and related model views plays a key role in supporting the increasing number of data exchange requirements.

The analysis of IFC is relevant in the context of this research because it reflects, in part at least, the state of the art in computational representation of buildings. Similarly to STEP, IFC is the result of a large, multi-national standardization effort that captures the current level of understanding and agreement among experts regarding the information needs of the AEC domain.

Given the scope, complexity and relevance of IFC, it will be reviewed separately in the next chapter. The chapter will also include some of the most recent efforts in the development of BIM applications and methods related to functional and behavioral aspects of design, mostly from a requirements specification and verification perspective. The chapter will end with a summary of the most important ideas developed during the evolution of building product models and integrated design environments, and will outline possible paths for research and development.

CHAPTER III

FUNCTION IN THE INDUSTRY FOUNDATION CLASSES (IFC)

The Industry Foundation Classes (IFC) is the largest and so far the most successful effort to date in the search for a comprehensive building model framework. IFC started in 1994, originally as an internal initiative of Autodesk aiming for the development of a set of integrated software applications. Soon after it became a major international effort involving the collaboration of several AEC industry partners. This expansion shifted the focus towards an open data model and neutral file format intended to support a wide range of life-cycle data exchange scenarios [112].

The new effort was initially coordinated by the International Alliance for Interoperability (IAI), a new non-profit organization set to manage the initiative. In 2005 the organization was re-named buildingSMART, increasing significantly its base of collaborators as well as the scope of its mission. Today buildingSMART is supported by a global network with around nineteen chapters in five continents. Its members include government agencies, software vendors, academic institutions, large architectural and engineering firms, building component manufacturers and construction companies among others. The IFC conceptual data schema and exchange file format became an ISO standard, registered as ISO 16739:2013. Currently IFC is on its fourth major version, i.e. IFC 4, and it is being increasingly adopted in the AEC industry worldwide [58].

The development of IFC capitalized on many of the lessons learned from previous experiences, particularly those under the STEP umbrella. However, while conceptually related to STEP mainly by the use of EXPRESS and EXPRESS-G modeling languages, IFC eventually became a separate effort. This was due in part to the recognition of the particularities of the AEC industry, and the unique set of challenges associated with the design, construction and operation of buildings.

An important idea to address some of those challenges was the use of process modeling

as method to capture the generation and exchange of data during different building project workflows. This method, first introduced in the COMBINE project, allowed to identify in a more systematic manner the different data requirements that arise during typical use-case scenarios. This provided a practical reference framework for the development of an information model general enough to cover most of the interoperability requirements found in the AEC industry.

In IFC, interoperability requirements are typically satisfied by subsets of the overall information model. This subsets are called 'model views', following the terminology of relational databases adopted in IFC [106]. A major advantage of the database view approach is that they can be explicitly defined in terms of the original schema, for example, in the form of pre-stored queries. The definition may include the use of rules for derivation of new data at the instance level, along with constraints to enforce its semantically validity.

Since the IFC model is essentially a relational schema, the use of model views allows the precise specification of IFC subsets required to satisfy the data requirements of a particular exchange. The process of specification for a model view in IFC involves two main activities. First, relevant building workflows need to be identified and mapped in the form of process models. This activity is carried primarily by AEC domain experts, aiming to provide an objective characterization of stakeholders, domain-specific applications and data exchange requirements involved in relevant workflows. This characterization is then documented in the so-called Information Delivery Manual (IDM), which provides the main reference for the specification of standardized model views [100].

The explicit, machine-readable specification of IFC model views is made using EXPRESS language, which allows conformance tests to be executed to ensure semantic validity within the overall IFC schema. An official model view specification, called Model View Definition (MVD), is then released to software vendors for the implementation of appropriate application interfaces and translators [326].

An important criterion within the IFC modeling framework is the reusability of patterns in the development of new IDMs and MVDs. In this way model extensibility is facilitated while semantic consistency is enforced through the adoption of best practices.

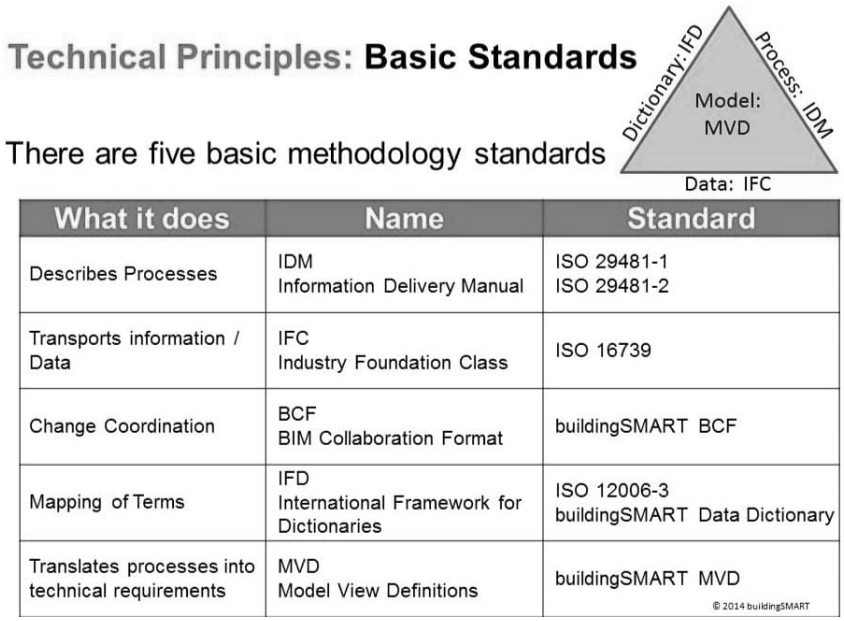

Figure 3.1: Basic Standards of buildingSMART OpenBIM initiative

.

For this reason both IDM and MVD are now part of the set of core standards promoted by buildingSMART (see Figure 3.1). Additional specifications included in this set are the buildingSMART Data Dictionary or bSDD (a reference library based on the ISO 12006-3 also known as International Framework for Dictionaries or IFD), and the BIM Collaboration Format (BCF). Along with IFC, these standards are intended to guide the implementation of BIM applications covering all aspects of the buildings life-cycle, from programming and design, to construction, operations, maintenance and demolition [59].

3.1 Geometric and non-geometric aspects in IFC

Broadly speaking, information pertaining to different aspects of the building life-cycle can be distinguished between geometric and non-geometric. In IFC however, most of the modeling effort has been devoted to the representation of geometric aspects. For instance, the representation of shape, location and the connectivity of various building systems and components, have been extensively covered in IFC. On the other hand, the coverage of non-geometric aspects has not received the same level of attention. Some of these aspects

are related to the representation of life-cycle requirements, from facility planning and architectural programming, to construction, operations and maintenance [202].

More recently, some of these non-geometric aspects have been addressed by efforts such as the Construction Operations Building Information Exchange standard (COBie) and the Building Programming information exchange standard (BPie). However, both standards rely extensively on information defined outside the scope of IFC. More specifically, they depend on external classification systems, such as OmniClass^{TM} and UniFormat^{TM}, which are developed mostly for human readability, and without formal semantics. The relationship between IFC, BPie and CoBie with external building classification standards is explained in the last version of the National BIM Standard - United States Version 3 [60].

It is important to note that the intent of covering the representation of design requirements in IFC is included in its very declaration of scope, as expressed in the official ISO 16739:2013 standard [188]. There, different BIM data exchange formats are presented as means to support integration during various phases of a building's life-cycle. These phases range from "demonstrating the need", the "conception of need", "construction", "operation and maintenance" [sic], to name just a few. It also includes in its scope more specific definitions such as "client requirements management" and "analysis items", among others. However, ISO 16739:2013 purposely excludes from its scope the definition of "behavioral aspects of components and other behavioral items' [sic].

This last part of the declaration of scope has a number of implications. First, it suggests that the representation of non-geometric aspects, such as functional requirements, can only be supported in IFC insofar they are treated in structural terms, i.e. using structural entities as proxy representations. As discussed in the previous chapter, this has been the traditional approach in the development of previous integrated design environments and building product models. Part of the reason can be attributed to the conceptual legacy of the problem-solving paradigm in design. From this perspective, the translation of functional requirements into structural constraints provided obvious practical advantages, from the viewpoints of both software implementation and operation .

Such pragmatic perspective is still prevalent today in IFC, as well as in several proprietary data models that support specification and verification of functional requirements. In the particular case of IFC, constraints may be defined using the entity *IfcConstraint* and applied to value properties of various objects. For instance, limiting values or boundaries that can be applied to single values properties [44].

A similar observation can be made regarding the purported 'analysis items' mentioned in the scope of IFC. In this case, properties are also used sometimes as implicit references to various behavioral aspects concerning performance. As in the case of constraints, this provides advantages at the implementation level, including the implementation of procedures for data extraction and verification. For example, analysis items regarding topology and placement of load bearing members (e.g. beams and columns) and connections can be derived from geometric properties of corresponding elements. This derivation provides an useful idealization to feed directly into analysis applications.

Nevertheless, such applications require access to additional analysis items which are not strictly structural, i.e. limited to topology, dimensions and placements. These additional items in fact characterize behavioral aspects that are either inherit in a design, or imposed during operational conditions. In the context of structural engineering analysis, these include behavioral properties such as loads conditions and associated set of actions and reactions [47]. Other analysis items may refer to thermal and acoustic properties relevant to performance evaluation applications.

Clearly, the use of such behavioral concepts in IFC indicates a contradiction to the aforementioned statement of scope. This contradiction seems further reinforced by the fact that certain types of processes are also declared within the IFC scope. Such processes defined in IFC under *IfcProcess* may refer to events, procedures and tasks for which various building elements can relate in different ways [43]. For each possible way, a specific behavioral aspect is indeed being described, although tacitly and in an ad-hoc fashion. For example, a mechanical equipment can be associated to various different processes, such as installation, maintenance and replacement. This mechanism would allow for example, the comparison and selection of equipment based on how well different equipment options perform under

each of these perspectives.

It is interesting to note however that the abstract entity *IfcStructuralActivity* used to define mechanical actions and reactions for structural engineering applications is considered in IFC as a subtype of product, i.e. *IfcProduct* and not as a subtype of process, i.e. *IfcProcess*. In general, these contradictions indicate a theoretical weakness in the conceptualization of IFC, including the semantic relationships that must hold among different ontological categories. The increasing demand for data integration for different types of performance evaluation has led in turn a rather idiosyncratic treatment of behavioral aspects, with several redundant and inconsistent definitions haphazardly spread across the entire data model.

This assessment will be discussed in more detail after a brief overview of the IFC schema.

3.2 Brief overview of IFC

The IFC model is organized according to four conceptual layers, each containing a particular sub-schema (See Figure 3.2). A first important distinction is made between the so-called rooted entities and the non-rooted entities. Rooted entities are all those entities that are defined as subtypes of *IfcRoot* under the Core layer. Non-rooted entities are those defined exclusively inside the Resource layer, and don't have a common super type. Resources entities cannot exist by themselves, but only by reference from rooted entities defined in the other layers.

3.2.1 *IfcRoot*

All building object definitions are derived from *IfcRoot* through multiple levels of inheritance. *IfcRoot* establishes Globally Unique Identifier (GUID) and other generic attributes to all its subtypes. *IfcRoot* has three direct subtypes:

- *IfcObjectDefinition*: Generalization of all object types and occurrences.
- *IfcRelationship*: Generalization of all objectified relationships.
- *IfcPropertyDefinition*: Generalization of all object properties.

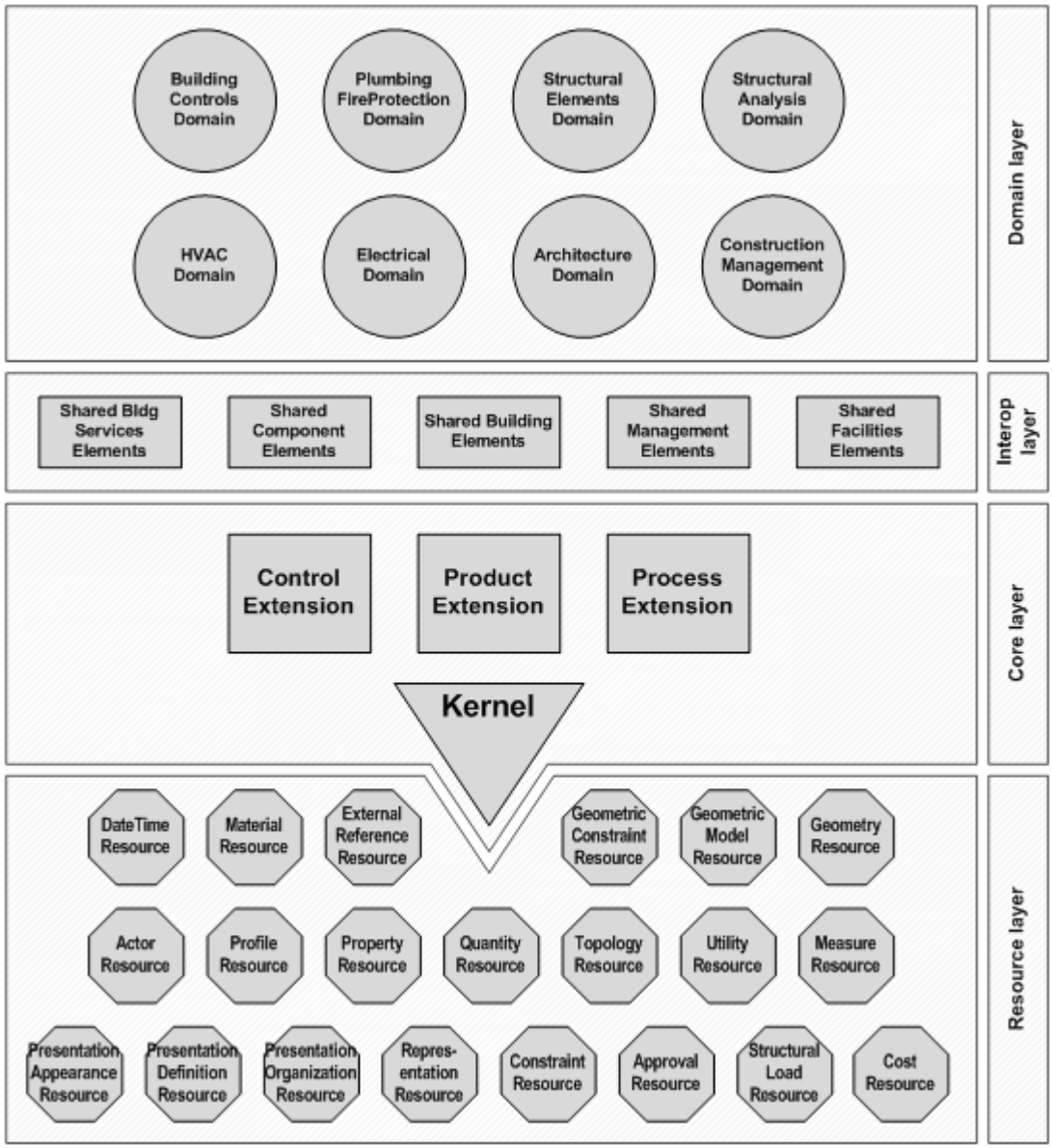

Figure 3.2: The main four layers of the IFC architecture. From buildingSMART [50].

3.2.2 *IfcObjectDefinition*

IfcObjectDefinition is described in IFC as the "generalization of any semantically treated thing or *process*, either being a type or an occurrence" [sic] [41]. *IfcObjectDefinition* has three direct subtypes:

- *IfcContext*: Generalization of a project contextual information.
- *IfcObject*: Generalization of all things that appear or occur in projects.
- *IfcTypeObject*: Specification of information about an object type to be assigned at the instance level (i.e. occurrence).

The ontological implications of the relationship between *IfcObject* and *IfcTypeObject* has been analyzed by Borgo in [30], and will be discussed later in subsection 3.2.4. Meanwhile, the focus stays on the definition of *IfcObject*, which has six direct subtypes:

- *IfcActor*: For the representation of stakeholders, either individuals or organization.
- *IfcControl*: For the representation of restrictions on products, processes or resources.
- *IfcGroup*: For the representation of collections of objects with a special purpose.
- *IfcProduct*: For the representation of design entities such as site, space, wall, etc.
- *IfcProcess*: For the representation of time-dependent concepts such as tasks, events, activities, etc.
- *IfcResource*: For the representation of entities subject to usage and limited availability, such labor and equipment.

3.2.3 *IfcRelationship*

A series of specializations of *IfcRelationship* allow different subtypes of *IfcObjectDefinition* to be mutually related. Figure 3.3 provides a graphical overview of the inheritance hierarchy of *IfcObjectDefinition*, along with the set of objectified relationships defined at the top level.

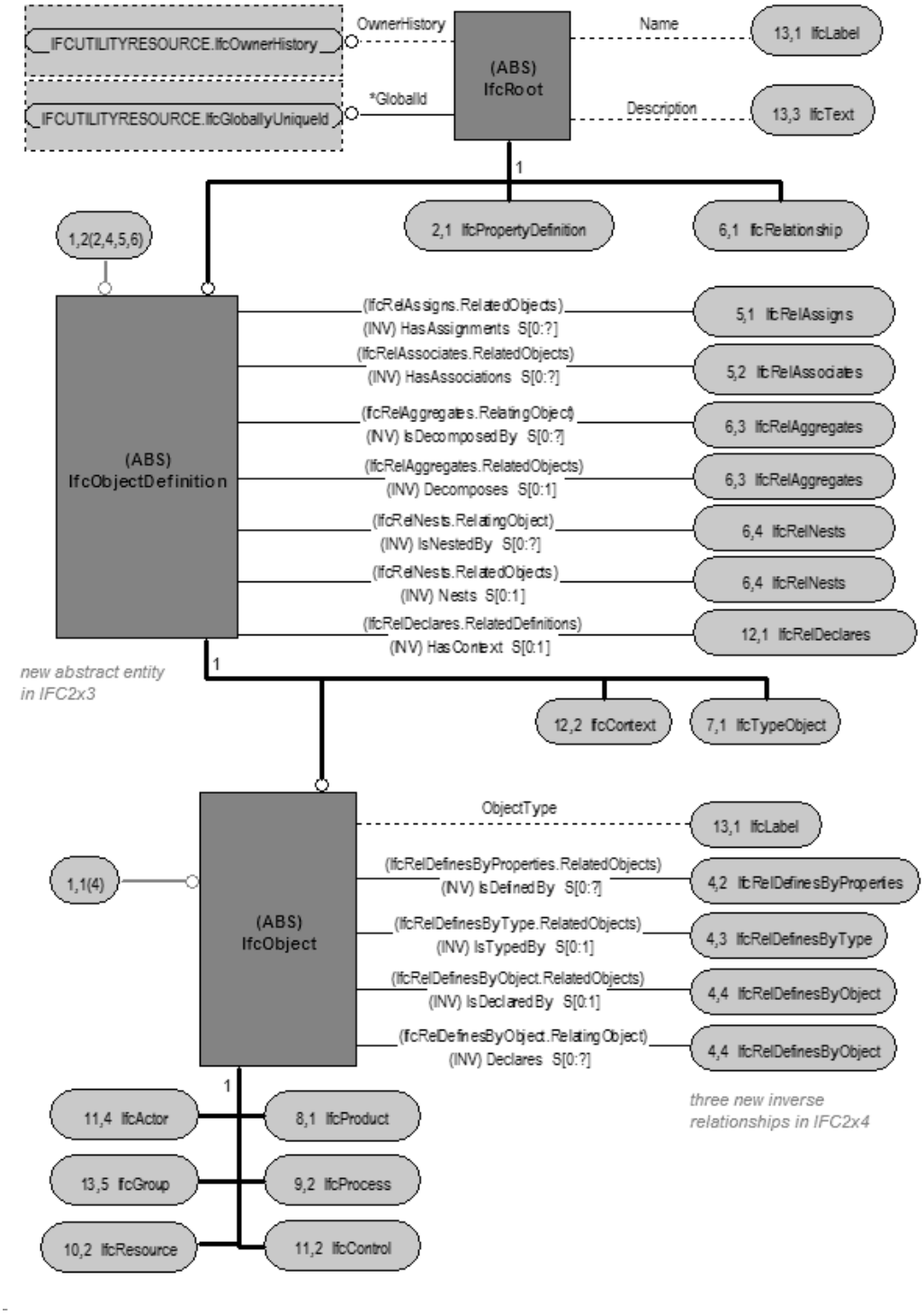

Figure 3.3: EXPRESS-G Diagram of top level IFC entities

Each relationship imposes different type and cardinality constraints over relating entities, as well as the possibility of other semantic validity rules. The following list provides a brief description for each:

- *IfcRelAssigns*: The assignment relationship is a generalization of "links" among object instances. It denotes the generic association between different objects, allowing for various forms of traceability (e.g. client-supplier).
- *IfcRelAssociates*: The association relationship refers to sources of information such as classification systems, libraries, or other documents, with no dependency implied.
- *IfcRelConnects*: The connectivity relationship connects objects under some criteria. It does not imply constraints, however subtypes of the relationship may apply constraints based on the semantics of the particular connectivity.
- *IfcRelDeclares*: The declaration relationship handles the declaration of objects or properties as part of a project or project library.
- *IfcRelDecomposes*: The decomposition relationship, defines the general concept of composition/decomposition, that is, a whole/part hierarchy. Subtypes may impose constraints over types as well as existential dependency between whole and parts.
- *IfcRelDefines*: A generic, abstract relationship. It allows further semantic characterization of IFC objects, most notably through property set definitions (e.g. *IfcPropertySetDefinition*) and object types (e.g. *IfcTypeObject*)

3.2.4 *IfcPropertyDefinition*

IfcPropertyDefinition provides the generalization of all characteristics that may be assigned to objects. Through this mechanism common property information about objects can be shared among all their subtypes and instances. *IfcPropertyDefinition* also provides the means for dynamically attaching new sets of properties to object occurrences.

The most relevant subtypes of *IfcPropertyDefinition* in the context of this analysis are *IfcPropertySet* and *IfcPredefinedPropertySet*. These are described as follows:

- *IfcPropertySet*: A dynamically extensible property set, for which IFC only provides only a basic "meta model", to be fully specified outside the standard. This means no definition for the properties involved exists within the IFC model. Type declaration is based on string matching of property names.

- *IfcPredefinedPropertySet*: a property set entity that exists within the standard IFC model. The meaning of each statically defined property set is given by its entity type, while the meaning of the properties is defined by the name and data type of the explicit attribute representing it.

As it will reviewed later, property sets allow, albeit implicitly, the functional and behavioral characterization of design entities through different mechanisms. Since such characterization of semantics is made by extension, it imposes limitations in the way that certain forms of automation could be implemented.

3.3 Functional and behavioral aspects in IFC

The observation that many IFC entity definitions indeed refer to different behavioral and functional aspects of buildings can be exemplified in the following group of inter-related entity definitions. Each of these will be analyzed separately in the following four subsections.

3.3.1 Processes and controls

A case where behavioral aspects of buildings are being indeed described explicitly in IFC is through the definition of *IfcProcess*. Processes are defined in IFC as *"actions taking place in a project with the intent of acquiring, constructing, or maintaining objects. Processes are place in sequence in time."*[sic] [39].

A typical scenario of use for this entity would be construction planning. For example, instances of *IfcProduct* (e.g. a scaffolding) and *IfcActors* (e.g. masons) can be assigned as resources (i.e. instances of *ifcResource*) to execute a particular construction task (e.g. laying bricks). Tasks in IFC (i.e. *IfcTask*) along with events (i.e. *IfcEvent*) and procedures (i.e. *IfcProcedure*) are all considered subclasses of *ifcProcess*. Subclasses of *IfcProcess* may be mutually related by a series of nesting relationships through the use of *IfcRelNest*, or

sequence relationships, using *IfcRelSequence*, which together establish a partial temporal order.

The entity *IfcControl* in turn allows for the specification of restrictions on any product, resource or process involved. These restrictions may take the form of action requests, cost schedules, permits and work calendars, among others. For example, an *IfcWorkCalendar* may be specified to indicate when an event is active [37], which in turn may be used to trigger an *IfcProcedure* [42].

In this way *IfcControl* acts as a form of requirement specification generally related but not restricted to construction planning and management. The objectified relationships *IfcRelNest* and *IfcRelSequence* in turn specified pre-conditions between processes required to achieve certain outcomes or objectives. Figure 3.4 exemplifies the relationship among tasks, events and procedures.

A possible use outside the scope of construction is the specification of distribution control elements such as sensors and actuators for building operation defined under *IfcDistributionControlElement*. These control elements may have control constraints specified by *IfcControl*, as well as associated procedures in case of a triggering event [35]. Distribution control elements typically operate over distribution flow elements (i.e. *IfcDistributionFlowElement*), such as storage, conversion or treatment devices intended to enable the distribution of energy or matter within, across or around buildings [36].

Altogether, *IfcProcess* and *IfcControl* provide a form of behavioral specification over building products that is fundamentally teleological. The reason why processes and process-specific relationships are not recognized as 'behavioral aspects' by the IFC standard is unclear. A more detailed discussion on this interpretation will be provided at the end of the chapter.

3.3.2 Products, groups and systems

The class *IfcProduct* is the most general entity definition for any object that can be described geometrically, and which is either manufactured, supplied or created on site as result of a construction process. Subclasses of *IfcProduct* share shape representation and spatial

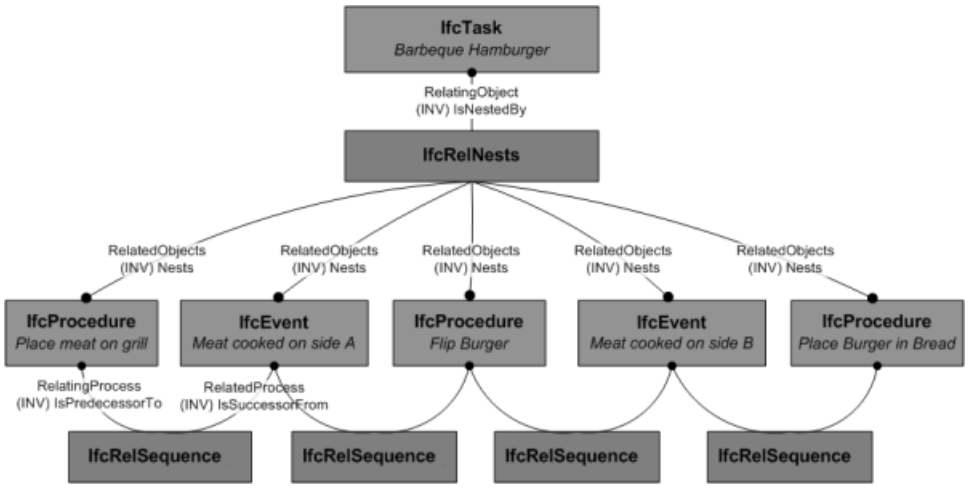

Figure 3.4: Example of relationships between task, event and procedure in IFC.

placement as main inherited properties. They also inherit a series of generic relationships such as aggregation, association and assignment. The latter relationship is used to assign products to processes as mentioned in the previous subsection. An *IfcProduct* can be a tangible thing, such as a roof, a door, or it can be intangible, such as a space (e.g. a room). This distinction is carried through two main subclasses of *IfcProduct*. Tangible products are comprised under *IfcElement*, while spatial ones are comprised under the general class of *IfcSpatialElement*. However, and less intuitively, an *IfcProduct* can also include virtual entities such as grids, ports, annotations as well as the so-called analysis items. These explicitly comprise behavioral descriptions such as physical actions and reactions mentioned in the previous section.

Instances of *IfcElement* and *IfcSpatialElement* can be grouped arbitrarily by assigning them to an instance of *IfcGroup*[38]. This assignment is made by means of the *IfcRelAssignsToGroup* relationship. One scenario where this may be used is to depict a collection of possibly dispersed parts that nevertheless work as a functional whole. This may include mono-functional technical systems or performance-driven aggregations. The latter would theoretically correspond to aspect systems.

However, only the first case receives treatment in IFC, through a series of specialized subclasses of *IfcGroup* corresponding to technical systems commonly used in buildings. These are defined under *IfcSystem*[48], which generalizes a range of distribution systems,

Table 1: Sample of distribution systems predefined in IFC.

Category 1: Pipe-based systems and ports	
CHILLEDWATER	Non-potable chilled water system .
DOMESTICHOTWATER	Heated potable water distribution system.
DRAINAGE	Drainage collection system.
FIREPROTECTION	Fire protection sprinkler system.
Category 2: Duct-based systems and ports	
AIRCONDITIONING	Conditioned air distribution system.
EXHAUST	Exhaust air collection system.
VENTILATION	Ventilation air distribution system.
Category 3: Cable carrier systems	
AUDIOVISUAL	Audio and/or video media stream system.
CONTROL	Network dedicated to control system usage.
EARTHING	A path for equipotential bonding, and current grounding.
POWERGENERATION	A path for power generation.

load-bearing systems, shading systems, envelope systems, and spatial systems.

For each of those, a set of optional predefined enumerated types is provided. Table 1 shows a sample of these enumerated sub-types available under *IfcDistributionSystem*, which inherit indirectly from *IfcGroup*. Distribution systems in IFC are classified under three main categories, depending if they are pipe-based, duct-based or cable-based.

The main criterion for this classification scheme seems to be the type of artifact in charge of the distribution, i.e. ducts, pipes or cables. Implicitly however, it is rather the type of physical element being distributed what drives the classification criterion. Thus, pipes are typically used for distribution of liquids of some sort, ducts are for gases, and cables are for electric energy, as well as electrical or optical signals. Each in turn implies a specific set of behavioral properties that need to be handled case by case by different applications. Velocity, pressure, temperature, amperage are some examples of relevant behavioral properties.

Even though many predefined groups like these are intended to describe conventional technical building systems with well-established functions, groups are not considered themselves as products in the IFC conceptualization. This is despite the fact that such groups are obviously the result of some manufacturing or construction process, and need to comply with specific performance requirements. This modeling choice seems problematic, since

much of the value of a building derives precisely from how such building systems are created and perform as a whole. Furthermore, a series of operational and maintenance costs are dependent on the life span of different components of a system, along with their level of integration. Despite of these practical consideration, the definition for *IfcProduct* does not offer any formal relationship to product functionality or performance at the component level that could be logically aggregated at the system level. Instead, references to functional or performance properties are provided on an as-needed basis by different subclasses.

There are two main methods in IFC by which this can happen. The first one is by means of so-called property sets. This is a mechanism by which properties can be defined independently and bundled together as sets. Such property sets can then be assigned to various object occurrences - including groups - whenever needed. For example, IFC groups, such as distribution systems, may have various domain-specific property sets assigned to them. Some may refer to flow-based properties like pressure, velocity, or temperature, to name a few. Many others can be defined and attached to multiple object types as well. Thus, property sets provide IFC with a flexible mechanism for semantic extensibility beyond the standard definitions of the schema.

The second method of functional or behavioral characterization is by means of association to external sources of information such as classification systems, libraries or legal regulations. This method is available to all subclasses of *IfcObjectDefinition*, including processes and groups. In general, functional and behavioral characterization using this method relies on textual descriptions that are not machine readable at all. Therefore, semantic interpretation relies exclusively on human judgment and subjectivity.

Property sets and external classification systems will be reviewed in the next two subsections.

3.3.3 Objects, object types and property sets

Property sets are intended to provide a flexible mechanism for semantic extensibility of IFC. It typically works by means of relationships with two classes of IFC entities, namely, *IfcObject* and *IfcTypeObject*.

The abstract entity *IfcObject* is defined as "the generalization of any semantically treated thing or process. Objects are things as they appear - i.e. occurrences." [40]. However, the information provided by *IfcObject* and its subclasses is not sufficient by itself to fully characterize a building element from all possible perspectives of interest.

To satisfy this need, IFC makes use of property sets defined under the entity *IfcPropertySet*, which allows properties to be defined independently and aggregated into bundles that can be shared by multiple objects. Such assignment can be made directly to all occurrences of an IFC object, or indirectly. The first alternative is made through the relationship *IfcRelDefinesByProperties*, which is depicted on the left side of Figure 3.5.

The second alternative is by assigning property sets to another class of entities called *IfcTypeObject* [49]. This approach allows various objects to be typed by means of the relationship *IfcRelDefinesByType* depicted at the top center of Figure 3.5.

In this way, multiple occurrences of *IfcObject* get further characterized by properties that are relevant from a particular point of view. Because of the contingent nature of property sets, such approach is especially useful in the specification of Model View Definitions (MVD) [100].

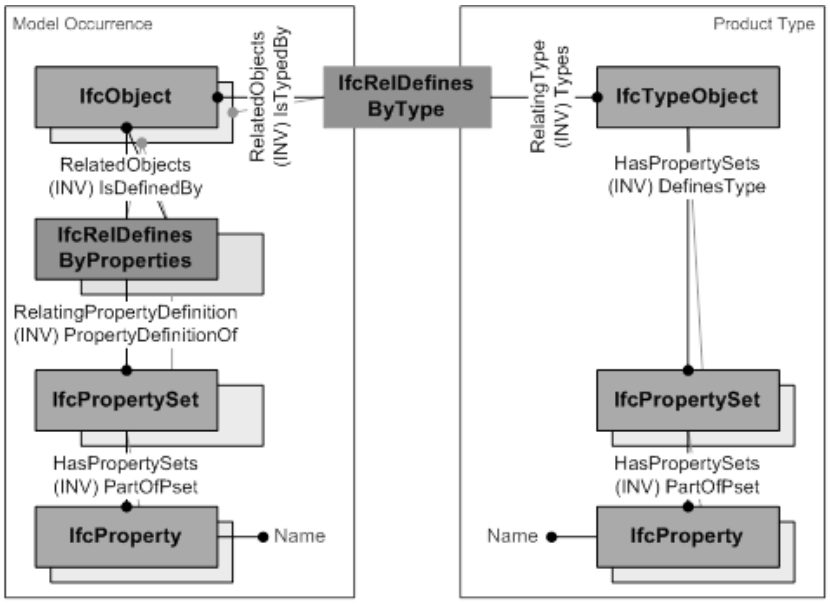

Figure 3.5: Relationship *IfcObject*, *IfcTypeObject* and *IfcPropertySet*. Source: buildingSMART.

The distinction between *occurrence classes*, as represented by *IfcObject*) and *type classes*, represented by *IfcTypeObject* is a key principle behind the development of the IFC data model. This principle is based on the so-called "*generic - specific - occurrence* modeling paradigm", which relies on property sets as primary mechanism for semantic extensibility [49].

The ontological implications of property sets in the semantic characterization of IFC has been analyzed by Borgo et al. [30]. In this study, property sets are discussed as potential means to provide *intensional* meaning to entity definitions. This would be necessary to provide formal treatment to the use of *extensional* definitions that constitute the core of the IFC taxonomy. The problem lies on the fact that the semantics of extensional definitions rely on the explicit reference to individuals, i.e. the set of instances, sharing some common property. In this scenario, the meaning of a class of entities depends directly on the existence of individuals that exemplify such class. The precise ontological nature of a common property however does not need to be made explicit.

However, the reliance on extensional definitions is problematic, since the criteria for taxonomic classification of occurrences might be ambiguous [168]. For example, Borgo refers to the problem of identifying whether an occurrence represents a physical entity with spatio-temporal location, or it stands for a representational artifact [30]. An example of such representational artifact in IFC is *IfcProxy*. This entity is a sub-type of *IfcProduct*, inheriting all properties related to spatial placement and geometric representation of products. However, it is primarily intended as informational wrapper or container for various types of objects, which may or may not have a geometry and spatial location. Thus, *IfcProxy* is a virtual entity mainly defined to facilitate data exchange between applications [45]. The same can be said of other virtual items under *IfcProduct*, such as grids, ports and annotations, as well as the so-called structural activities, which in IFC denote physical actions and reactions of load-bearing structural members.

Intensional definitions on the other hand do require the common property of occurrences to be made explicit. Typically, this is a complex property made of the conjunction of all relevant properties that characterize a class type [30]. In this way the semantics defined

by intension do not depend on the existence of occurrences, but on the specification of necessary and sufficient conditions for class membership. In particular, this involves the specification of certain types of relationships that must hold with other entities at the same level of abstraction.

Thus, the distinction between extensional and intensional definitions is ontologically relevant because certain properties are considered essential to the identity of an entity. For this reason, the formalization of such properties would be considered a main criterion for taxonomic classification within a certain domain [231, 167, 168].

Conversely, many properties are contingent to particular information requirements, and therefore are not essential to the core definition of an entity. Unfortunately, such distinction among properties is not recognizable in the definition of property sets, whether they are assigned directly to *IfcObject* or indirectly via *IfcTypeObject* occurrences.

The general case of walls is particularly illustrative. For instance, the load bearing capability of walls and the state of being external, i.e. part of the boundary separating the interior from the exterior of buildings, are both attributes defined within the property set *Pset_WallCommon*, which is applicable to all occurrences of walls [57]. Both attributes however can change over the lifetime of a wall, without such wall ever ceasing of being a wall. The case where the wall might be external is irrelevant for its identity. Indeed, such wall may be first conceived as an external wall, by setting the Boolean property *IsExternal* as (TRUE). Later, an addition to the building can make the wall to become internal, i.e. by setting *IsExternal* as (FALSE). The same change may occur to the load bearing status of the wall, assuming that the meaning of 'load-bearing' is restricted to loads applied along the plane of the wall.

On the other hand, it is possible to argue that most essential property of a wall is to partition adjacent spaces. The goal state of completely bounding a space is just a particular form of partitioning, achieved either by a single continuous wall or by the combination of multiple walls.

Yet, mere physical separation of spaces cannot be considered in itself an essential property of walls either, since it only establishes a structural characterization, namely, the

demarcation of boundaries between adjacent spaces. A similar result could be achieved by simply drawing lines on the ground. But even then, a behavior is implied, since an intelligent agent could be taught to recognize the lines as symbols, and to act accordingly to their intended purpose [277].

From a Building Design perspective, a stronger behavioral characterization is required, so that walls can be clearly differentiated from other types of spatial boundaries. Such characterization should refer, first and foremost, to the behavioral capability of walls to work as physical barriers against flows of different kinds trying to move across them.

From this initial behavioral characterization, other behavioral properties can be established in relation to the type of physical entity to be retained or obstructed, and how the retention or obstruction need to happen. For instance, walls can be designed to keep entities as diverse as heat, dirt, moisture, noise, animals or even criminals, to name a few, from physically crossing their boundaries. The destination of the crossing, i.e. inbound or outbound, is less important, since different entities need to be retained on either side of the wall boundary. The function of *allowing* crossing or access through the wall is provided in turn by different types of wall openings. In this context, doors and windows are special devices placed in wall openings to allow extra control over the direction and rate of flows.

Hence, the load-bearing capability described in the property set *Pset_WallCommon* indicates an additional level of functionality that is not essential to walls. But this observation is only true if load conditions are restricted to gravitational and lateral forces acting along the plane of the wall, and not across the plane. For loads applied in the normal direction of the wall, the analysis is very different. In fact, the argument presented here is that it is precisely the capability to resist loads in the normal direction what constitutes the most essential property of walls. From this basic capability, the retention of different types of entities and flows becomes possible. The fact that a partition wall is only able to resist a tiny fraction of the loads of a (soil) retaining wall, does not negate its *retaining* capability at all. Otherwise, people simply could not lean themselves or other objects such as paintings against the surface of partition walls.

Because of this, partition walls are also subject to the similar failure modes associated

to retaining walls, albeit less frequently. These include overturning, flexural failure, tilting and sliding from their base. Nevertheless, it would be misleading to conclude that the main difference between partition and retaining walls is just a matter of degree regarding loads.

Instead, the nature of the agents exerting the loads, along with the phenomena associated to their agency, provide an ontological basis for their differentiation. Thus, layers of soil, including water content, are the primary agents acting on a retaining wall. The primary function of soil retention is achieved by keeping the soil and its contents, from crossing the physical boundary of the wall. Besides withstanding horizontal pressure, a retaining wall often requires other behavioral capabilities, such a providing water drainage to to dissipate hydrostatic pressure. In different scenarios, retaining walls need to be water-tight.

A partition wall on the other hand, deals with very different types of agency. Typically these involves people, animals, or artifacts, with diverse sets of associated phenomena to be retained and dissipated.

In the case of IFC however, there is no reference to the general retaining function of walls. The only minor reference to that capability seems to be provided by the item SHEAR, part of the enumeration *IfcWallTypeEnum*[55]. According to the official description, a SHEAR wall is a wall "*designed to withstand shear loads. Such shear walls are often designed having a non-rectangular cross section along the wall path. Also called retaining walls or supporting walls they are used to protect against soil layers behind.*"[sic]. While such description may suggest that load-bearing capabilities in the normal direction are concomitant to retaining walls, a further note in the specification page seems to indicate otherwise. In particular, the note indicates that "*the potentially misleading term SHEAR shall not impose a particular resistance against shear forces, but a particular shape.*" [sic] [55].

This last note imposes an inconsistency regarding the identity criteria for retaining walls, since just few sentences above in the same document, the description refers to the retaining capability of walls by mentioning other alternative names, and by referring to the goal of protection against soil layers. Yet, according to the latter note, the only property that really matter seems to be structural, that is, the shape of the wall.

The implication is relevant, because the general capability of retention in the normal

direction is what defines a wall at its most fundamental level. This capability encompasses a conjunction of behavioral properties, including, but not limited to (withstanding) some degree of shear in at least one direction. From this perspective, to add the word 'retaining', as in 'retaining wall' seems redundant, since all walls are built to fulfill a general function of retention.

Instead, in order to be more explicit with the intended meaning, the term 'soil-wall' could be used, in the same way that the term 'firewall' is used to indicate the fire-*retaining* function.

Similarly, the same general retaining function is also essential to curtain walls. This can be verified by the fact that curtain walls are especially designed to retain or restrict access to different types of entities from the external environment. What differentiates curtain walls from other partition walls however is not the separation relation of internal and external spaces, as it is often assumed, but how curtain walls stand in place in first place. Whereas conventional partition walls stand in place by bearing on structural elements immediately below, curtain walls hang from structural elements from above. Hence, the identity criteria that differentiates the latter from the former is also fundamentally behavioral. In other words, the relationship between compressive versus tensile behaviors of their internal structure. In this context, the geometric location of curtain walls is not essential to their ontological identity.

This distinction offers the advantage of eliminating unnecessary restrictions on designers, given that in curtain walls can be placed practically anywhere, insofar they stand in place by predominantly hanging from a supporting element above them. In the most common case, where curtain walls are placed in a building, the location can be at boundary between indoor and outdoor spaces. Similarly, the supporting element may or may not be the roof structure itself.

Although IFC does not formally restricts curtain walls to be external, nor that they should only be hanged from the *edge of the roof or roof structure*, this assumption is implied by the description provided in the official documentation of *IfcCurtainWall*[52].

In the example provided in Figure 3.6, there are two internal curtain walls separating

the central atrium from two triple-height lobbies on each of side. The atrium is totally enclosed within the building. While the curtain wall on the left is in fact hanging from an edge above, this edge does not belong to the roof, but to a slab on the fourth floor of the same side. Meanwhile, the curtain wall on the right is hanging directly from the roof structure, but not from its edge, but from mid-span of supporting girders. Therefore, none of the properties described by *IfcCurtainWall* are essential for the identity of curtain walls, in the same way that they are not essential for any wall in general.

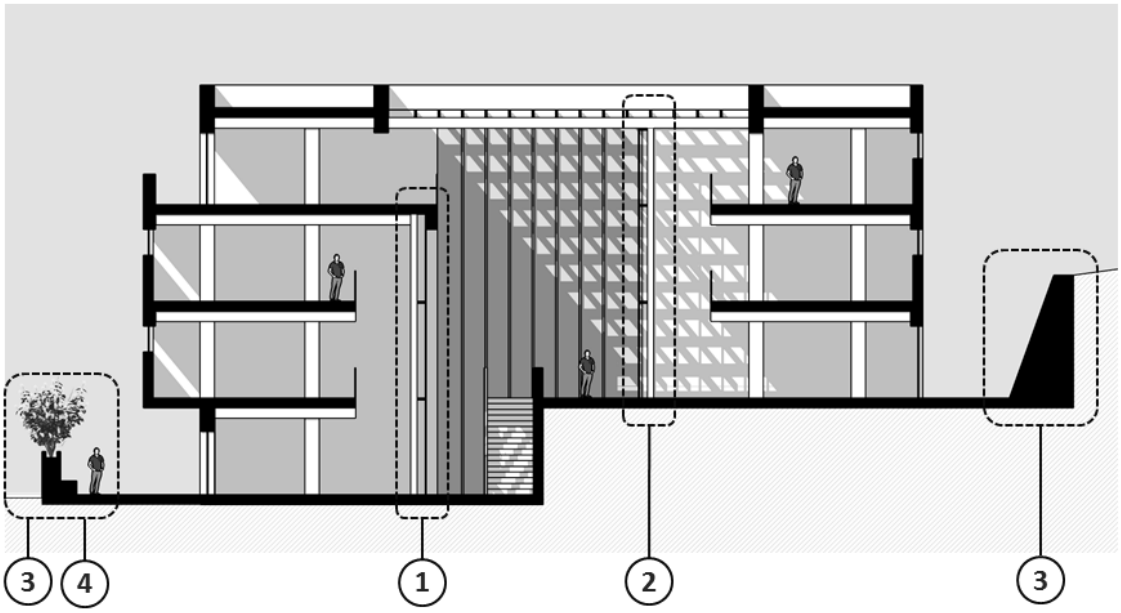

Figure 3.6: Two internal curtain walls facing an internal central atrium. 1) The curtain wall on the left hangs from the edge of a slab on the third floor. 2) The curtain wall on the right hangs from midspan of rooftop girders. 3) Extension to the structure above is irrelevant for the retaining wall and parapet outside of the building. 4) Parapet on the left also serves as planter and bench.

Despite sharing the same *retaining* function as essential property, walls and curtain walls are extensionally defined in IFC as disjoint classes. Interestingly, if property sets are taken as basis for intensional characterization of semantics, i.e. as complex property, a very different interpretation can be made. Indeed, the analysis of property sets for both wall types indicates not only that they are not disjoint, but that actually the entity wall may be considered as a specialization of curtain wall, and not the other way around as it could

Table 2: Comparison of property sets in IFC for walls and curtains walls.

Pset_WallCommon	Pset_CurtainWallCommon	Description
Reference	Reference	Reference ID for this specified type in this project.
Status	Status	Status of the element, predominately used in renovation or retrofitting projects.
AcousticRating	AcousticRating	Acoustic rating for this object. It is provided according to the national building code.
FireRating	FireRating	ire rating for the element. It is given according to the national fire safety classification.
Combustible	Combustible	Indication whether the object is made from combustible material (TRUE) or not (FALSE).
SurfaceSpreadOfFlame	SurfaceSpreadOfFlame	Indication on how the flames spread around the surface.
ThermalTransmittance	ThermalTransmittance	Thermal transmittance coefficient (U-Value) of an element.
IsExternal	IsExternal	Indication whether the element is designed for use in the exterior (TRUE) or not (FALSE).
LoadBearing	N/A	Indicates whether the object is intended to carry loads (TRUE) or not (FALSE).
ExtendToStructure	N/A	Indicates whether the object extend to the structure above (TRUE) or not (FALSE).
Compartmentation	N/A	Indication whether the object is designed to serve as a fire compartmentation (TRUE) or not (FALSE).

be assumed. Table 2 provides a comparison for both property sets *Pset_WallCommon* [57] and *Pset_CurtainWallCommon* [56].

It can be seen how the properties are practically the same all the way down to property *IsExternal*. This means not only that their names are the same, but also their value types and textual descriptions provided in the official Documentation (IFC 4 Addendum 2) are the same. After property *IsExternal*, the property set *Pset_WallCommon* provides three additional properties, namely *ExtendToStructure*, *LoadBearing*, and *Compartmentation*, none of which, as discussed before, are intrinsic to the general class of walls.

Since the properties of 'being hanged from', and 'extending to a structure above', are not formally defined in *IfcCurtainWall* nor in its associated property set, nothing formally

forbids a curtain wall occurrence of actually not hanging at all. In such situation, a occurrence may be considered a curtain wall specialization supported by compression from below, which leads to a partition wall.

In general, inconsistencies like these have to be handled by users at the instance level, as result of the principle of extensionality adopted in IFC. This principle is further embraced in the development of the ISO 12006-3 / International Framework for Dictionaries (IFD), promoted by buildingSMART as one of its core set of standards (see Figure 3.1. An important aspect of the overall buildingSMART standardization effort is the association with external construction classification systems used by the AEC industry.

While these systems are mostly intended for human readability, there is an expectation that some of them will contribute as conceptual framework for the future development of IFC and related standards. Among these external systems, the OmniClass^{TM} Construction Classification System (OCCS) is especially relevant. Since the concept of function is a key criterion for the organization of OCCS tables, the relationship between IFC and OCCS is analyzed in the next subsection.

3.3.4 External associations

Different specializations of *IfcObjectDefinition* may be associated to external sources of information, such as libraries, product catalogs, regulations and approvals, as well as external classification systems. This concept of association is an important principle towards a comprehensive building modeling framework as originally envisioned by IFC. Through the link with external data sources many critical aspects of the building life-cycle can be covered beyond the immediate scope of IFC. For the specific the case of classification systems, IFC provides the relationship *IfcRelAssociatesClassification*.

To promote this type of association throughout the industry in a more uniform way, the National BIM Standard-US recommends the use of a series of interrelated information standards. Among them, OmniClass^{TM} is recommended as the main classification system for all classes of building entities to be used by BIM workflows in North America [60].

The OmniClass^{TM} Construction Classification System, also known as OCCS, is developed and maintained by the Construction Specification Institute (CSI) in collaboration with the Construction Specification Canada (CSC) association. OmniClass^{TM} is designed to provide a standardized basis for classifying information created and used by the entire North American AEC industry, covering the full life-cycle of facilities, from inception to operations and maintenance, to demolition and reuse. Harmonization with other international efforts has also been considered a key aspect of its development, mostly by following the standard ISO 12006-2: *Organization of Information about Construction Works - Part 2: Framework for Classification of Information* [82].

Another guiding principle behind OCCS was the compatibility with legacy systems such as Uniclass, MasterFormat^{TM} and, UniFormat^{TM}, also developed by CSI, among others. Eventually, OmniClass^{TM} will evolve into a umbrella framework covering most of the concepts addressed by these other legacy systems, albeit at a more general level.

Because of its broad coverage, and as part of a larger international effort for standard harmonization, it is expected that the continuous development OmniClass^{TM} terminology will play a key role in the evolution of IFC [82].

The fact that the concept of function is one of the main criteria for terminological classification in OCCS, a preliminary analysis of this system is provided here. However, only few OCCS definitions are reviewed in order to exemplify the association between IFC entities and OCCS from a functional perspective. A more detailed review of OCCS is offered in Appendix A.

3.3.4.1 OmniClass^{TM} and related standards

The OmniClass^{TM} classification system (OCCS) is predicated on the ISO 12006-2: *Organization of Information about Construction Works - Part 2: Framework for Classification of Information*. According to its most current version, the ISO 12006-2:2015 provides a general framework for the development classification systems for the built environment. For that purpose, it identifies a set of recommended classification table titles for a range of different information object classes. These tables are formulated according to particular views of the

Table 3: Comparison of tables in ISO 12006-2 and OmniClass™

OmniClass™ Tables	**ISO 12006-2 Tables**
OmniClass™ Table 11 - Construction Entities by Function.	ISO Table 4.2 Construction entities (by function or user activity). ISO Table 4.3 Construction complexes (by function or user activity). ISO Table 4.6 Facilities (construction complexes, construction entities and spaces by function or user activity).
OmniClass™ Table 12 - Construction Entities by Form	ISO Table 4.1 Construction entities (by form).
OmniClass™ Table 13 - Spaces by Function	ISO Table 4.5 Spaces (by function or user activity).
OmniClass™ Table 14 - Spaces by Form	ISO Table 4.4 Spaces (by degree of enclosure).
OmniClass™ Table 21 - Elements (includes Designed Elements)	ISO Table 4.7 (by characteristic predominating function of the construction entity). ISO Table 4.8 Designed elements (element by type of work).
OmniClass™ Table 22 - Work Results	ISO Table 4.9 Work results (by type of work).
OmniClass™ Table 23 - Products	ISO Table 4.11 Construction products (by function).
OmniClass™ Table 34 - Organizational Roles	ISO Table 4.15 Construction agents (by disciplines).
OmniClass™ Table 35 - Tools	ISO Table 4.14 Construction aids (by function).
OmniClass™ Table 41 - Materials	ISO Table 4.17 Properties and characteristics (by type).
OmniClass™ Table 49 - Properties	ISO Table 4.17 Properties and characteristics (by type).

entities being addressed, e.g. by form or function of products or systems at various stages of the facility life-cycle.

The ISO 12006-2 is not intended to provide a complete operational classification system, nor does it provide the content of the tables. Instead, it provides a general framework to be used by organizations developing and publishing such classification systems around the world. In this way, by sharing a common framework, it is expected that harmonization between local implementations of ISO 12006-2 would be greatly facilitated [189].

Table 3 shows the tables titles defined in ISO 12006-2 and their interpretation in OCCS.

It is important to point out that classification systems based on ISO 12006-2 are primarily intended for human consumption, in the sense that they are entirely text-based, with no formal, machine-readable semantics. The same observation applies to other legacy systems developed by CSI and CSC, such as UniFormat™ and MasterFormat™.

To address in part this limitation, a second related standard, the ISO 12006-3: *Organization of Information about Construction Works - Part 3: Framework for Object-oriented Information* was initiated. The original purpose of ISO 12006-3 was to specify an object-oriented framework to support the development not only of classification systems, but also associated product and process models [187]. Later on, ISO 12006-3 merged with similar efforts such as the STABU LexiCon initiative from Holland, and BARBi from Norway, becoming known as the International Framework for Dictionaries (IFD) [33]. Currently, IFD is developed under the leadership of buildingSMART, and it is considered one of its five basic standards [59].

The ISO 12006-3 / IFD standard relies only partially on the organization principle of tables developed in ISO 12006-2. Instead, it focuses on entries of the tables as primary source for an object-oriented representation of construction entities. In this approach, entities can be described by properties or characteristics without a predefined grouping or hierarchical classification[83]. This is intended to facilitate extensibility of existing schemes, as well as flexibility in the way different entities can be related at the instance level. For example, in situations where view-dependent aggregations are required.

Clearly, this approach is intended to leverage and complement the principle of *extensionality* adopted in IFC, in an attempt to cope with the increasingly dynamic and unpredictable nature of data requirements. In other areas, such approach is been addressed by the use of dynamic programming languages, and the reliance on abstractions such as *duck typing*. This mechanism is intended to provide dynamic typing capabilities over rigid class hierarchies, enabling greater software adaptability in contexts where use case scenarios and data requirements change continuously, e.g. web applications [242]. However, at the time of writing this dissertation, no evidence was found regarding the use of duck typing for dynamic processing IFC entities.

Furthermore, given that IFD is largely an international effort, dealing with complex issues of harmonization across different information systems, it remains to a large extent, a work in progress. Meanwhile, OCCS has been adopted as the primary construction classification system and the main reference for construction terminology for BIM applications

and workflows in North America [60].

Even thought OCCS is primarily intended for human readability, there are number of ontological issues in its classification criteria that may eventually hinder future efforts for data integration, interoperability and automation. Some of these will be reviewed next, particularly in relationship with IFC.

3.3.4.2 Association between IFC and OCCS

Regarding IFC, there are three main types of IFC entities for which associations with OmniClass^{TM} are relevant from a functional perspective. These are *IfcBuildingElement*, *IfcSpatialElement* and *IfcMaterial*. There are other IFC entities which also may benefit from such external associations. These include entities such as *IfcGroup* and its subtypes defined under *IfcSystem*. However, as discussed in the previous subsection, groups and systems are not considered as products themselves in IFC. For this reason the present analysis will concentrate exclusively on the first three entity types.

The entity *IfcBuildingElement* is one of the main subtypes of *IfcElement*. It comprises all tangible products that are considered a '"major functional part of a building". These include elements such as walls, roofs, floors, etc. which are often assumed to have a single main function, but may have others. Tangible building elements are made of materials which can be described under *IfcMaterial*. Spaces defined under *IfcSpatialElement* are either surrounded by instances of *IfcBuildingElement* or may contain them.

For further specification, instances of *IfcBuildingElement* can be associated with various tables of OmniClass^{TM}, such as OmniClass^{TM} Table 23 - Products, OmniClass^{TM} Table 22 - Work Results or OmniClass^{TM} Table 21 - Elements. Instances of *IfcMaterial* in turn can be associated to Table 41 - Materials. Spaces can be associated to Table 13 - Spaces by Function. IFC entities can also be associated to OmniClass^{TM} Table 41 - Properties, for further specification of objects characteristics.

The association to particular OmniClass^{TM} entries is based on an unique identification number provided by OmniClass^{TM}, making it amenable for database implementations. Yet, it is important to notice that the mechanisms for association differ significantly between

IFC products such as building elements and spaces, on one hand, and IFC materials, on the other. The former requires the use of the *IfcRelAssociatesClassification* relationship, while the latter relies on *IfcExternalReferenceRelationship* [53, 54]. This is an important distinction, since *IfcBuildingElement* and *IfcSpatialElement* are defined as rooted, first-class entities, while *IfcMaterial* is defined as a non-rooted entity. Non-rooted entities, as explained earlier, belong exclusively to the Resource layer schema (See Fig 3.2). For this reason they can only exist at the instance level as dependent properties of rooted entities. In other words, materials are considered as second-class entities that in principle cannot exist by themselves in the IFC conceptualization.

Such would be the case even for concrete entities such as bricks and insulating boards. A priori, these can only be referred to at instance level as part of a set of masonry wall layers (e.g. through the usage of *IfcMaterialLayerSet* by an instance of *IfcWallStandardCase*). In this way, bricks and insulation boards cannot exist independently with their own geometric representation and location outside of the wall that instantiate them. To work around this, IFC provides an alternative wall entity type to depict more complex wall assemblies (i.e. *IfcWallElementedCase*). However, this approach has been found limited for many situations where specific instances of bricks and insulation boards need to be represented as full-fledged first-class products [26, 68].

Furthermore, this assumption regarding dependency however is problematic from a functional perspective. This is because the functionality of products depends to large extent on the materials they use, and the behavioral characteristic they afford. The same applies, albeit indirectly, to spatial elements, since they are enclosed by products and the materials they are made of.

For example, the acoustics of an auditorium is as much the result of its geometry and dimensions, as it is the result of the behavioral properties of the materials chosen for its walls, ceiling, floor and furniture. Therefore, from a functional point of view, it can be said that products, including spaces, *depend* on the materials they use, and not necessarily the other way around. To be more precise, the foam used in an acoustic panel does not depend on the panel for its existence. Arguably, pieces of foam material could be added

haphazardly to an auditorium as improvised solution to fix some acoustic problems. Hence, the only dependent entities involved are the acoustic properties of the material, and not the material itself.

This perspective implies the need for a rather different type of dependency relationship between materials and products that the one assumed by IFC. A similar argument could be made regarding the geometry of products, which is also defined as a dependent entity in IFC. However, there is an important ontological difference between materials and geometry. Materials, for the purpose of design, refer to entities that extracted, manufactured or procured from third sources. They also have to be stored, transported and handled on site under certain conditions similarly to any other type of building product. These requirements imply labor and energy costs along with various environmental impacts. None of these aspects can be applied to geometry alone, which is essentially a mental abstraction.

Moreover, materials can take different geometric forms during their life-cycle, being their manifestation in a particular product only the last of a series of successive trans-*formations*. In this sense, what distinguishes their temporary manifestation as a product from others manifestations is the intentionality of design.

The position that materials are first-class entities with an independent existence from the so-called products is made explicit in OmniClass$^{\text{TM}}$. This is apparent in cases where materials are used either as ingredient for another material or used in their original state to achieve some work result. Such entities are defined in OmniClass$^{\text{TM}}$ both as material as well as product. For example, sand and bricks are treated in this way. Curiously, concrete blocks are considered only as a product, the same as mortar and grout [83]. Table 4 provides a sample of material entries with associated definitions. Notice that material composition, origin, function and origin are used haphazardly as definition criteria. More examples are provided in A.

This interpretation is not only in conflict with the one adopted by IFC, but it also adds another level of inconsistency regarding the nature of parthood or composition relationships. Similar ontological problems arise regarding the definition of living elements used for landscaping and construction. For instance, climbing plants such as vines used in fences

Table 4: Sample of Material from OmniClass^{TM} Table 41

Number	Title	Definition
41-10 10 60 11	Aluminum	Atomic Number 13; alloys are used primarily in windows, doors, cladding.
41-30 10 25 19 19	Sand	A naturally occurring compound made of rock and mineral particles.
41-30 10 25 13 11	Brick	Typically bound with mortar, brick blocks are a common building product used to construct nearly every type of vertical structure.
41-30 30 11 19 13 11	White Oak	One of the most pre-eminent hardwoods of *eastern North America*.

and building screens may be considered either as a material, a product, or as work result. Rammed earth used in wall construction provides another similar classification challenge.

These and other classification issues remain to a large extent unresolved in OmniClass^{TM}, as they are in IFC. As consequence, redundant definitions, misplaced hierarchical relationships and semantic inconsistencies are common in both.

The lack of proper *use-mention distinction* seems to be at the root of many these classification problems [287]. This typically happens when different contexts of meaning are treated as they were the same [286]. In the particular case of OCCS, this problem occurs when aspects of reality that are supposed to be the subject matter of the classification (e.g. physical or performance properties) are confused with abstractions adopted to describe them (e.g. systems of representation and measurement).

This issue is particularly problematic in the classification of ''Performance Properties" under Table 49 - Properties. This category is intended to describe a variety of behavioral aspects that are subject to measurement and control. These include concepts such as Reverberation Time, Air Tightness or Glare Index, to name a few. However, it also includes the so called Testing Properties (Number 49-81 11 00). These include Test Methods, Test Authority, and Inspection Protocol. Generally speaking, the latter concepts are concerned with the administrative processes of specification, verification and validation of behavioral performance, rather than being performance properties themselves.

Finally, the category "Performance Properties" also includes a sub-category Tolerance Properties (Number 49-81 15 00), which includes a series of dimensional tolerances, that

Table 5: Sample of Performance Properties from OmniClass^{TM} Table 49

Number	Title	Definition
49-81 00 00	**Performance Properties**	Properties that express the behavior of an object in reaction to physical properties and forces.
49-81 11 11	Test Method	Describes the type of test to be performed.
49-81 11 13	Test Authority	Designates the originator of the test or the standard to which the test results are held.
49-81 11 21	Inspection Protocol	Rules and procedures governing inspection and investigation of systems or work.
49-81 15 21	Squareness	Shaped like a square.
49-81 21 31	Weatherability	Resistance properties for heat, moisture, cold, solar radiation, etc.
49-81 31 11	Adhesion Strength	Measurement of the amount of force a bonding agent can withstand.
49-81 81 13	Reverberation Time	The persistence of sound in a particular space after the original sound is removed.

otherwise should be considered structural and not behavioral. Table 5 provides a short sample of these different entries under Performance Properties.

In summary, these inconsistencies hinder the ability of OmniClass^{TM} to support the organization, sorting and retrieval of information as its stated objective. Moreover, the lack of a formal criteria for the definition and classification of construction information restricts the potential of OmniClass^{TM} for supporting the future development of IFC and related information technologies.

3.4 Discussion on IFC and related efforts

The analysis of IFC is relevant in the context of this research because it reflects, in part at least, the state of the art in computational representation of buildings. Similarly to STEP, IFC is the result of a large, multi-national standardization effort that captures the current level of understanding and agreement among experts regarding a domain of knowledge.

In the case of IFC, the agreement is driven by the information needs of the AEC industry, in relation to different stages of a building project life-cycle. A similar observation can be made regarding classification systems such as UniFormat^{TM} or OmniClass^{TM}. Even though

these systems cannot be considered machine-readable, they are intended to complement and inform the development of IFC and other related machine-readable standards, by defining shared vocabularies and taxonomies.

While vendors of BIM applications typically develop their own proprietary information models, it is reasonable to assume that the analysis provided in this chapter can also be applied to them. There are two reasons for this belief. First, the development of information models and classification systems typically follow the conceptualization *perceived* as the most accepted in practice. This assumption may be considered true even when such conceptualization has aspects that remain problematic. Second, there are business advantages for developers to maintain certain level of conceptual alignment or harmonization with open standards such as IFC and OmniClass^{TM}.

3.4.1 Classification criteria: function or structure

Because of the aforementioned reasons, and the highly pragmatic goals of the building industry, there is no evidence that theoretical aspects related to functional representation are treated any different in the development of proprietary software applications. Figure 3.7 presents the two general classes of walls offered in Autodesk Revit, a popular commercial Building Information Modeling application. In Revit, structural walls refer to walls with load bearing capabilities, while architectural walls generally refer to walls with no such capabilities, e.g. partition and curtain walls.

Historically though, there has never been such a distinction in the context of Architecture. Indeed, structural walls have always been considered architectural entities. The difference at hand is mainly one of functional specialization and not identity. Thus, structural walls *are* architectural walls that play a load-bearing role, in addition to other functional roles common to most architectural walls. Among these, the role of providing a vertical barrier to flows of different kind.

Despite the restrictive and misleading terminology, the basic mapping of these two classes of walls in Revit into IFC is relatively straightforward, given the generic nature of wall properties defined in IFC (by means of *Pset_WallCommon*). At the geometric level,

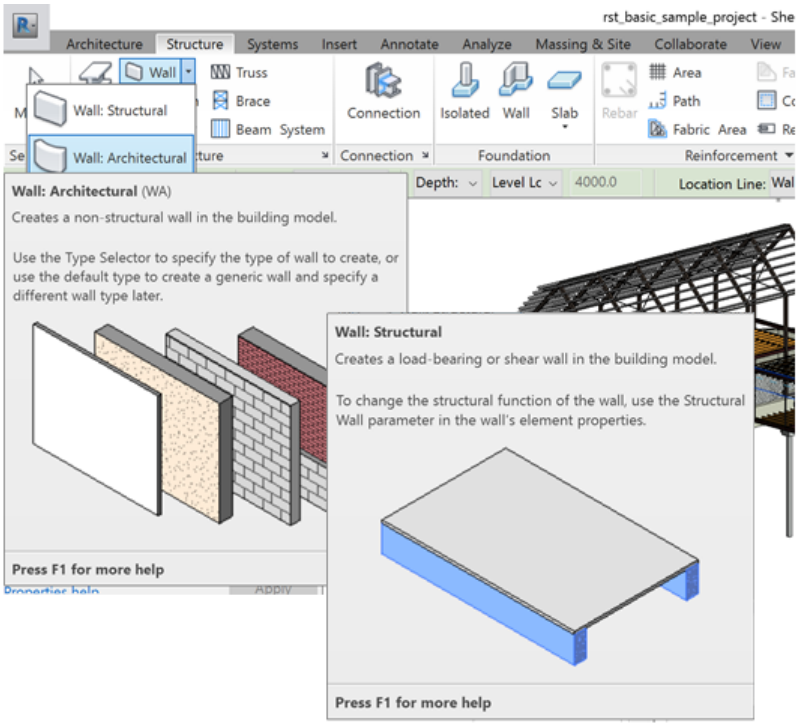

Figure 3.7: Basic wall types in Revit. Description for architecture walls on the left, structural walls on the right.

more specific geometric constructs are invoked, depending on the information requirements of a particular design task or exchange. For instance, idealized geometric or topological representation for the analysis of load-bearing members.

Revit and other BIM applications also support direct association to external classification systems, including, but not limited to OmniClass^{TM}, without necessarily relying on IFC. Thus, design entities can be informally characterized from a functional or behavioral perspective using the text-based taxonomies and dictionaries provided by third parties.

3.4.2 Structural surrogates

Regarding machine-readable processing though, the representational resources available are far more limited. In general, the prevailing approach still remains very similar to the one adopted in the earlier design environments and building product models, reviewed in Chapter 2. This involves for the most part the use of structural properties as representational surrogates of behavioral phenomena. As discussed earlier, this provides a convenient method for requirement specification and rule-based, parametric verification of constraints.

For example, structural properties of single building spaces (*IfcSpace*), or space groups (*ifcZone*), such as area or space location, can be automatically checked against particular occupancy or accessibility requirements [202, 214]. Other implicit property values can be derived by means of simple arithmetic calculations prior verification of compliance [292]. These may include structural properties such as glazing-to-floor ratios [98], or behavioral properties such as sound power ratios required for assessment of acoustic performance [249].

The derivation of more complex properties requires the generation of auxiliary data structures or models, especially when such properties characterize not a single IFC entity, but an aggregation of diverse entity types. This applies for example in the generation of topological graph models for automatic verification of circulation rules within IFC building models [219, 191], or in the detection of safety hazards in construction sites [332]. In all these cases however, structural features of the model are used as representational surrogates to provide a derived interpretation of the behavioral phenomena of interest.

For instance, the geometric intersection between restricted and security circulation spaces inside court houses denotes an undesirable possibility of prisoners and judges crossing their paths, with all the negative behavioral implications associated. Similarly, the intersection between geometric representations of holes in slabs or walls with circulation areas used by construction workers denote an undesirable possibility of safety hazards (i.e. falling accidents).

The computation of structural adjacency and intersection relations using graph-based or geometric-based operators are also used for the detection of other possible behavioral conflicts. These may include the detection of physical collisions or spatial interferences. One of the most common applications for this method in Building Design is the detection of spatial or physical clashes between different building subsystems. For example, between parts of the duct system with other elements such as walls, beams and columns.

It is important to point out that in this particular context, the structural representation of a 'detected' intersection is a physical impossibility, since two physical bodies cannot occupy the same space at the same time. Hence, the geometric clash is just a representational proxy intended to denote a more complex set of behavioral phenomena with procedural

implications. For instance, that some set of construction or assembly processes cannot be executed as specified by design. Such a situation usually requires design modifications, with possible delays and cost increases, among other unintended consequences [322, 321].

In both cases above, the use of structural properties as representational surrogates for behavioral phenomena is certainly convenient from both a cognitive as well as computational perspective. This is because it avoids the cost of describing an entire set of causal relationships, along with a higher computational cost. Unfortunately, the behavioral meaning originally intended is often lost overtime, especially when there is no formal, intensional characterization for the meaning of the relationship.

3.4.3 Behavioral surrogates

The use of representational surrogates is not limited to structural properties however. Behavioral properties, such as the thermal resistance or sound absorption capability of building elements, to name a few, are also used. In this case, a similar observation can be made, since these properties do not hold a formal, explicit characterization in terms of their causal relationship to the phenomena being denoted either. Such a causal relationship is most of the times tacitly processed in the head of human experts, and not in the models.

Conventions of practice, especially within domains, reinforce this problem by often assuming causality as trivial, and therefore not requiring explicit description. The negative consequences of this assumption emerge whenever high-level coordination among different design domains and construction trades is required. Here, the mere detection of intersections, clashes and other undesired structural relationships is just a first step for the far more difficult task of resolving underlying behavioral conflicts, and for which limited computational support exists.

Figure 3.8 illustrates two examples of clashes detected in a BIM application (Solibri Model Checker). On the left a pipe is clashing with a so-called 'architectural wall', that is, a non load-bearing partition wall. On the right, the pipe clashes with a 'structural wall'. Again, this classification is based purely on administrative criteria, in order to delineate professional ownership over different parts of the model (e.g. read and write privileges). It

has nothing to do with the functions themselves.

As discussed earlier, structural walls *are* architectural walls with added load-bearing capabilities. Since architectural walls can also have other less obvious functions, the criteria used to judge the severity of clashes is both idiosyncratic and tacit, depending on the user's perspective and value system. Consider for instance a not so hypothetical case where the non structural wall on the left is used to hang a very valuable painting. In such situation, the severity of the clash would be deemed very differently, assuming that such intent was communicated explicitly.

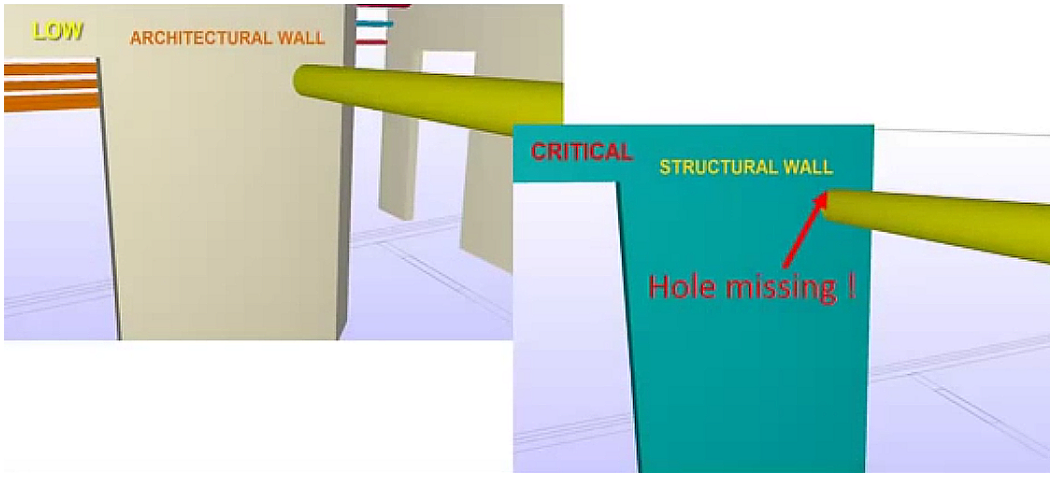

Figure 3.8: Clash detection in Solibri Model Checker. Severity of clash between pipe and architectural wall deemed as 'low' (on the left). Severity of clash with structural wall deemed as 'critical' (on the right). Rationale for severity levels based on user's perspective. Extracted from video online [291].

In this general context, the use of representational surrogates alone, such as those provided by geometric intersections, have little to offer. This is because representation surrogates only provide an extensional characterization of entities based on the existence of instantiated values. While this is useful for parametric verification of normative constraints (e.g. automatic rule-checking), it leads to semantic *dead ends* that lack functional information required for high level coordination and conflict resolution. According to Minsky, a number cannot reflect the considerations that formed it [234]. The same limitation applies

to other property value types, on which parametric verification of constraints relies.

3.4.4 The need for performance-driven functional aggregations

Part of the problem stems from the way functions and expectations of performance are expressed and represented. In all the theoretical models and applications analyzed in this research, two main recurring approaches for functional representation have been identified. The first one is primarily text-based, while the second approach is based on property constraints. While in the second case, automatic verification have been implemented with relative success, in neither case a formal, machine-interpretable representation has been provided.

Hence, issues related to change propagation and teleological consistency remains fundamentally unresolved, as none of these methods support the mapping of non-geometric inter-dependencies across multiple aggregation hierarchies [112]. In other words, what different parts are supposed to do, and how they can possibly interact with each other from different points of view.

This limitation is particularly important from a performance standpoint, where the satisfaction of functional requirements according to specific, quantifiable measurements demands careful integration of different systems. With the increasing interest in the industry towards performance-based design and performance-based building codes, the reliance on constraints as representational surrogates will no longer suffice. For that purpose, the identification of all parts from different and seemingly unrelated subsystems having functional participation will require a more sophisticated and robust functional modeling approach.

The aggregation of all these parts conforms the aspect system of the functional / performance requirement [16]. In principle, the general concepts of group in IFC, defined by *IfcGroup*[38], and its specialization *IfcSystem*[48], seem to share that goal, as generic containers for objects related to some common purpose or function.

However, the computational difficulty lies on how to traverse a design model so to harvest only those parts playing a significant role in the achievement of a given performance. For that purpose a machine-interpretable description of causal behaviors is needed, as criterion

to guide model traversal and harvesting of parts with functional participation. If design entities are also described in similar terms, then more robust and flexible mechanism for selection and aggregation could be implemented, including the possibility of automatic inference. Unfortunately, IFC does not provide the level of semantics required to support this type of capability. The same observation can be extended in general to building models used by proprietary applications.

As result, functional groups, or other forms of functionally-driven aggregations have to be either prescribed according to a normative view (e.g. Model View Definitions), or generated through ad-hoc methods. The former approach requires strict typing of design instances for tight coupling with target analysis applications. However this approach severely restricts flexibility in the structure of workflows, with potential impact in the design itself. In general, this approach is only appropriate for late stages of design, or for cases of repetitive design, where workflows are fixed and innovation is less of a priority. The second approach on the other hand, usually requires some form of ad-hoc dynamic property matching during model traversal, potentially leading to false positives or other types of semantic inconsistency.

In the case of IFC, the quasi-formalization of entity definitions hinders the development of a more comprehensive solution. Among the issues identified by this research is the lack of intensionality in the semantic characterization of entities, as pointed out by Borgo et al. [30].

At another level, the problem also relates with the informal, and often tacit characterization of functional and behavioral aspects in IFC. This observation includes references and resources for the description of functional requirements, processes and actions, which sprawl haphazardly in different parts of the IFC schema [202]. While the definition of scope in the preamble of the IFC standard clearly states the exclusion of such aspects, the analysis provided in this chapter demonstrates that such is not the case.

Indeed, the fact that design entities are intended to serve a purpose, and that they are evaluated based on their *behavioral* capabilities to accomplish such purpose, cannot be easily avoided from a product model conceptualization. While the challenges of providing

an operational representation of functions and behaviors are certainly difficult, the omission does not facilitate the treatment of the larger problem. Instead, it only adds complexity and unnecessary confusion, by leading inevitably to patches, workarounds and other ad-hoc solutions. The use of external classification systems, such as OmniClass, while necessary, also reinforces the negative consequences of idiosyncratic definitions and references.

3.4.5 Towards an operational representation of functions and behaviors

The critical challenge so far has been to provide a comprehensive formal vocabulary for the description of functionality, from both the 'demand' as well as the 'supply' side of design. This means a shared, machine-readable representation for the specification of required functions, i.e. functional requirements, and the specification of functional roles played by different artifacts spanning their entire life-cycle. Given such representation, different mappings between needs and solutions could be supported, eventually contributing to the process of systems integration under multiple performance criteria.

Regarding machine-readable specification of functional requirements, the need for ontological approach has been preliminary discussed for different areas. These include building performance evaluation [18, 16], automated code-checking [331, 249], and prefabrication in construction [313, 70], among others. Unfortunately, no concrete direction has been provided yet. In order to do so, two inter-related sets of issues require attention for the development of an operational framework.

The first one is to address the need for a compatible characterization of different types of function, as they apply to socio-technical systems such as buildings. In particular, it is necessary to make explicit the relationship between functions at multiple levels of abstraction. More specifically, this involves the clarification of the relationship between global, high-level building functions, which are not necessarily technical or physical, with low-level functions that tend to be not only technical, but also domain-specific. Traditionally, this relationship has been assumed to be a composition / decomposition relation. Unfortunately, this assumption has not proved to be successful, as it reduces the description of building functions to single hierarchies without accounting for the emerging, lattice-like structure of

functional dependencies.

The second issue has to deal with the relationship of functions and behaviors with structural entities. In this regard, two options can be considered: they can be treated either as proper, first-class entities, or as relational properties. The latter means that they stand as objectified relationship between structural entities, and other first-class entities which account for the phenomena of interest, namely, the set of processes, actions and events playing the functional role intended by design.

Currently, neither of these options can be observed in conceptualization of IFC, proprietary building models or external classification systems reviewed. As discussed in this chapter, functional and behavioral aspects are referred to using value properties as proxies. These include primitive value types such as string, integer, real and logical, or simple enumeration types that lack the level of semantics required to capture the relational nature of functional inter-dependencies that emerge across different building subsystems.

In order to address this problem, it is necessary to reconcile the demand and supply dimensions of functions. For that purpose, it is useful to consider a particular function as a subset of all possible behaviors that are intended by a stakeholder, i.e. the demand side, or exhibited by a system, i.e. the supply side. Given that the demand of a function usually requires certain behavioral pre-conditions to occur at a lower level (e.g. low-level functions), such behavioral pre-conditions characterize the behavioral space of the demanded function. Conversely, the function supplied by a system can also be associated to a behavioral space, which is characterized by all behaviors exhibited by the system under different conditions of use.

The intersection of both behavioral spaces indicates the participation of the system in the satisfaction of a demanded function. Notice however that such participation does not need to be a positive one. For instance, a behavioral pre-condition for the satisfaction of acoustic comfort is the avoidance of noise. In other words, noise is part of the behavioral space of acoustic comfort. On the other hand, the functioning of a HVAC system (e.g. air diffusers) may cause hissing as side-effect. Thus, hissing is part of the behavioral space of the HVAC. This denotes an overlap with the behavioral space associated to acoustic comfort,

that can be inferred by subsumption of hissing under the general category of noise.

The goal of this research is to provide this type of inference automatically, so that all entities playing a role in the satisfaction of a function can be identified and formally represented as part of the aspect system of the function. The following propositions summarize the modeling criteria. They also provide a preliminary road-map for the implementation of a proof-of-concept.

1. **On the definition of design entities (i.e. structural entities).**

    1.1. Structural entities need to be defined by intension, according to an ontology of behaviors.

    1.2. Intensionallity derives from the conjunction of behavioral properties that are essential to their identity.

    1.3. A conjunction of behavioral properties denotes the *behavioral space* of a structural entity. More specifically, they denote an initially assertion regarding the most common behaviors typically associated to a building element.

    1.4. Initially asserted behavioral spaces could get further expanded by the context in which the design entity is intended to operate.

    1.5. The context is provided by the set of functional requirements, in relationship to other design elements considered as part of the overall system.

2. **On the specification of required and supplied functions.**

    2.1. Required functions, need an ontology-driven, model-based, specification, i.e. a requirement model.

    2.2. A requirement model, denotes the *behavioral space* of the required function.

    2.3. The *behavioral space* denoted by a requirement model provides an initial context for the identification of functional roles, as well as for evaluation of their performance from multiple perspectives.

3. **On the elucidation of aspect systems.**

   3.1. The intersection of the two aforementioned behavioral spaces implies the possibility of a structural entity playing a functional role in the satisfaction of a functional requirement.

   3.2. At first, functional roles do not need qualification, in terms of being positive or negative, or how much positive or negative.

   3.3. Multiple structural entities with functional roles in the satisfaction of a same functional requirement, belong to the aspect system of that functional requirement. Parthood here implies behavioral interactions within the aspect system.

   3.4. A structural entity playing a functional role in satisfaction of multiple functional requirements, belongs to the corresponding aspect systems of each functional requirement involved. Multiple parthood in different aspect systems implies functional inter-dependencies among aspect systems involved, with potential conflicts requiring resolution.

The next chapter provides a brief historical review of the different theories and models developed in the fields of Artificial Intelligence and Applied Ontology. This review informs in part the propositions above. Specific theories and models underpinning goals and methods of this research are discussed at the end, leading to the formulation of the hypothesis and methodology.

CHAPTER IV

FUNCTION IN ARTIFICIAL INTELLIGENCE

The search for a formal, machine-readable representation of teleology started in the field of Artificial Intelligence by the end of 1960's and early 1970's [239, 238, 281]. Initially, these efforts focused on problem-solving as defining characteristic of intelligent behavior. Promising results soon motivated the first attempts to explore the suitability of AI theories and models in the area of design, insofar design could be treated as a type of human problem-solving activity [101, 136].

Later new approaches were explored, focusing on the representation of commonsense knowledge and the development of systems capable of qualitative reasoning about physical phenomena (i.e. naive physics) [171, 89, 213, 90, 23, 197]. Eventually other key dimensions of human cognition were brought into account, including the roles of memory in analogical reasoning and learning [141, 124].

The integration of analogical reasoning with model-based representations provided a new line of research, with significant potential to support various aspects of engineering design. Among them, the ability to solve new design problems by reusing known solutions from similar problems stored in memory was considered especially relevant. For that purpose, machine-readable descriptions of design problems were required, so that they could be used as indexes for searching relevant precedents or analogues. These descriptions in general took the form of teleological models grounded on different vocabularies, schemes and domain ontologies of structure, function and behavior [278, 3].

Model-based representations of function and behavior were introduced under an analogical design framework, intended to support a variety of design situations, and in service of different design tasks, including problem formulation, generation, evaluation, modification and validation of design candidates. [159, 71, 150, 153]. In this regard, an important focus of those efforts was on the development of an explanatory component or rationale

for the reasoning process, so that the working principles of a design could be explained in qualitative terms [134, 123, 3], and design failures could be diagnosed [198, 74, 158].

Regarding the field of application, the traditional target of most AI efforts in functional representation and reasoning has been in the area of product development, specifically of mechanical and electrical devices [308, 303, 212, 71, 76], although some research also focused on aspects of Building Design [142, 258, 251, 319, 225].

Parallel efforts, both in theory and practice of engineering design were proposing more systematic approaches to deal with the challenges of product development in the context of increasingly competitive global markets. In this area, methods of functional decomposition took a predominant role, providing a framework for communication and coordination of concurrent engineering activities to tackle multiple life-cycle requirements in shorter time-to-market spans [247, 297, 299]. However, few practical applications have been developed based on formal methods of representation and reasoning developed in AI [118].

One the main challenges is the diversity of meanings of the terms 'function' and 'behavior' commonly found in engineering design practice, and the lack of agreement in finding a common, operational definition that could satisfy most interpretations and use case requirements. More recently, collaborative efforts with the area of applied ontology have been exploring the possibility of an unified notion that could make different interpretations of function ontologically compatible. The proposed unification is intended to cover not only technical functions of designed artifacts, but also idiosyncratic functions ascribed to objects by users, as well as biological functions [235]. Eventually this will provide an ontological foundation for the development of design applications dealing with integration of biological and socio-technical systems, for which Building Design is an example.

Before dealing with some of the current developments on the ontology of function, the dissertation provides an historical overview of the main theoretical approaches and models developed in Artificial Intelligence. The goal is to recognize conceptual frameworks, common trends and limitations that could contribute towards a more comprehensive theoretical foundation for this research, and its potential contribution to the area of Building Information Modeling.

4.1 Early teleological models

This section offers a review of early research in Artificial Intelligence, and the development of computational programs that attempted to simulate intelligent problem-solving behavior. In general, resolution of problems took the form of a goal state, to be achieved through a sequence of inference steps involving composition of operators and state transitions. Promising results motivated the applicability of this approach in design, were goals took the form of functions to be achieved by composition of structures with associated behavior. However, approaches for the representation of teleological knowledge, as well as methods of teleological reasoning varied considerably. This variation reflects an evolution regarding ontological commitments adopted by different models, with increasing levels of conceptual coverage and sophistication. In particular, it reflects a shift from the initial problem-solving perspective towards other cognitive aspects of human intelligence, related to memory, learning and qualitative, commonsensical reasoning. Inference with partial or incomplete knowledge and abductive reasoning capabilities for hypothesis formulation, explanation of causality and rationale justification were all issues gradually introduced in the evolution of functional representation models. Most of these remain relevant areas of research.

4.1.1 General Problem Solver (GPS)

The General Problem Solver (GPS) was a computer program developed by Newell and Simon at Carnegie Mellon University, as part of a larger research effort to formulate a psychological theory of human problem-solving [238]. In this context, GPS was an attempt to simulate cognitive processes behind human problem-solving, with the goal of providing not only a theory, but also a computational model capable of generalization.

One of the main insights behind GPS was the recursive structure of the problem-solving process. This means that complex goals could be decomposed into series of smaller, more manageable sub-goals. In particular, each goal / sub-goal is represented as a state, to be achieved through the invocation of a proper sequence of actions or operators. For that purpose, a priori knowledge about states was provided (i.e. a knowledge base), allowing a solver to evaluate the differences between them, and to select the best operators required

to transition from one into another.

To constraint the search of the solution space, GPS made use of two types of heuristics, namely means-ends analysis and planning. The first was used for the decomposition of goals into sub-goals, while the second allowed to find a solution in simpler, more general terms through abstraction [239].

The principle of separation introduced by GPS, between domain-specific knowledge bases and logical solver justified the idea of generalization of the model. This in turn opened an entire new field of research, especially in the areas of engineering and architectural design.

It is important to note that besides its implementation in GPS, the Means-Ends Analysis (MEA) technique also played a major influence in various design methods. In particular, various approaches for requirements engineering based on functional decomposition can be traced back to MEA, including some discussed in chapter 2.

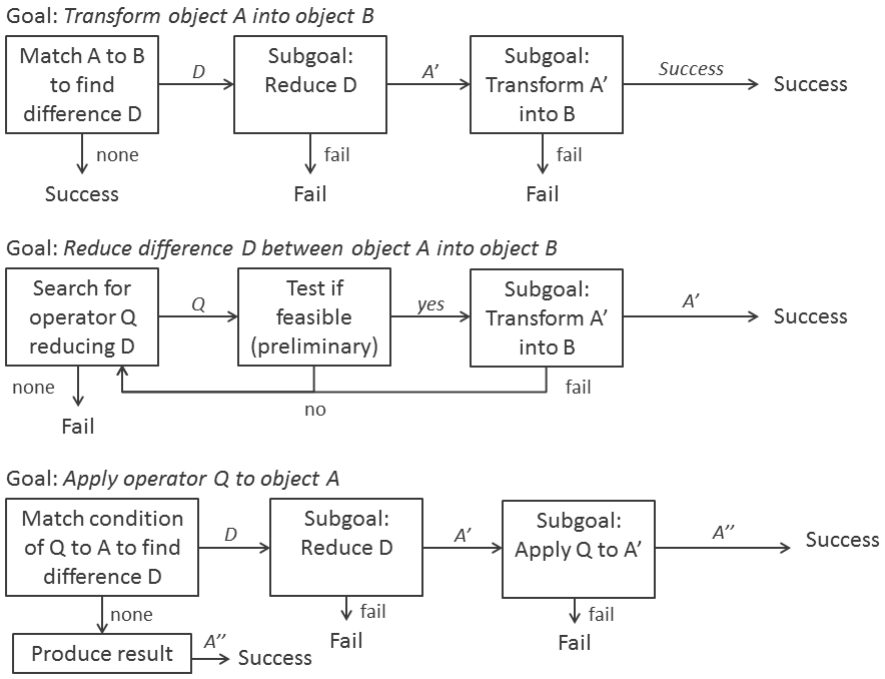

Figure 4.1: Recursive goal decomposition in GPS using Means-Ends Analysis. Modified from [238]

Given initial promising results in the so-called 'toy worlds', interest emerged in applying

the problem-solving approach to design problems. In this context, Freeman and Simon developed a qualitative model for functional reasoning in design based on GPS. Very soon the challenges of providing complete and consistent definitions of function became apparent, as well as the limitations of single representation methods and reasoning procedures [136]. This entails the need for stronger methods grounded on a theory of knowledge representation that recognizes multiple dimensions or aspects of a problem.

4.1.2 STRIPS

Promising results in early computational problem solving led to an increasing interest in the applicability of these frameworks to deal with more complex challenged. Among these, planning was considered an especially relevant case, with potential use in a wide range of automation scenarios. The Stanford Research Institute Problem Solver (STRIP) problem solver, developed by Fikes and Nilsson was one of the most important initial efforts.

In STRIPS, the task of the problem solver was to find valid compositions of operators able to transform initial states of a world model into goal states. In particular, the units of composition became the plans themselves, represented as abstract aggregations (i.e. nodes) of both state models and sub-goals and This aggregation allowed STRIPS to take advantage of forward search using MEA as heuristics, and backward search using theorem proving algorithms [265].

For purposes of search and evaluation of operators, STRIPS represented them as a schema of preconditions and post-conditions. The latter consisted of the addition and deletion of clauses describing the new state of the world model resulting from the application of such operator. For this reason, post-conditions were used as indexing mechanism for searching and selection of relevant operators given a required goal or sub-goal.

Preconditions on the other hand were used to assess the applicability of selected operator given the initial state of the world model. In cases where a selected operator did not have its preconditions met by a model state, then such preconditions became a new sub-goal to be achieved at a lower level of the search tree [127].

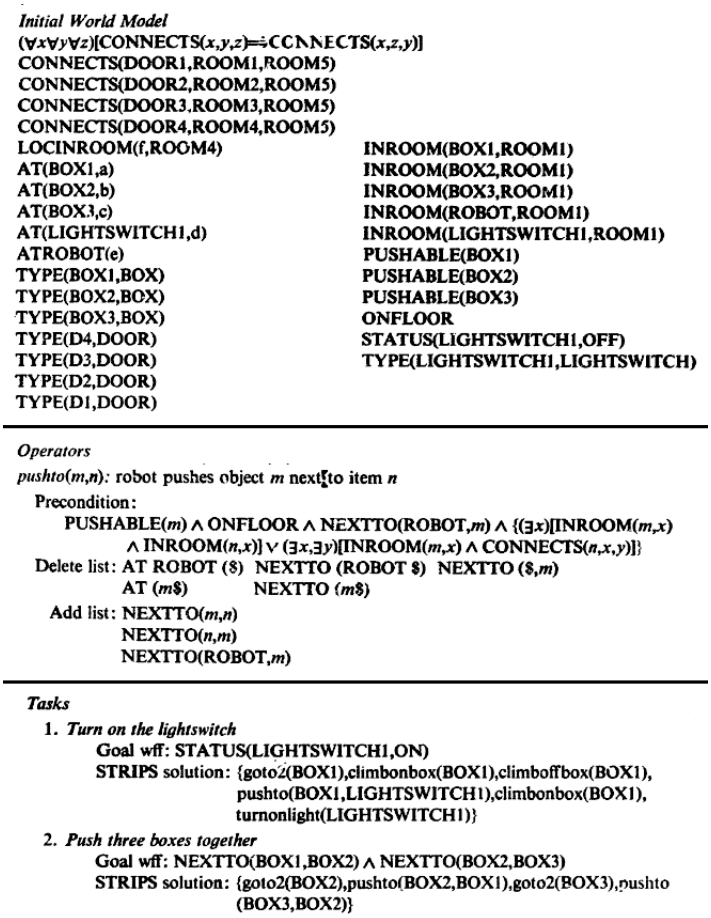

Figure 4.2: STRIPS world model and operators. From [127]

The logical evaluation of preconditions using theorem proving was key in enabling backward decomposition of plans. It is interesting to note that there are two types of preconditions admitted in the STRIPS ontology. The first refers to structural features of the environment, such as location in space or topological relations such as adjacency and connectivity.

The second type refers to behavioral capabilities or predispositions of structural entities, such as boxes that can be *pushed* around and *climbed* upon. These capabilities can also be referred to as *affordances*, following to the terminology introduced by Gibson [146].

In STRIPS, affordances of objects were either explicitly asserted in the world model, or implicitly assumed based on entity types referred to. In the example of the robot provided in [127], one of the preconditions for the robot to be able to push a box around is that such

box needs to be *pushable*. This is explicitly asserted by clauses like PUSHABLE(BOX).

On the other hand, for a robot to climb a box using operator *climbonboxm*, the instance *m* needs to be typed as a box. i.e. TYPE(m, BOX)). In this case, *climb-ability* is assumed as an affordance of the box, but this is not made explicit.

The fact that certain capabilities are explicitly assigned to objects while others are not, illustrates the underlying ontological challenge at the core of the problem of functional representation. If a robot could infer *climb-ability* out of the structural properties of boxes, then it certainly could infer *push-ability* as well. Both are potential roles of boxes, based on the effects obtained from certain types of interaction.

The need for more robust ontological formulations that address some of these teleological issues started to explored gradually, as new perspectives in human cognition and reasoning emerged. These included not only the description of causal relations in commonsensical terms, but also aspects of evaluation and modification of alternative solutions under conditions of intentionality.

4.1.3 Frames

It became increasingly more evident to AI researchers that resolution of complex problems required the integration of different types of knowledge, and therefore, different forms of knowledge representation.

In this regard, a comprehensive framework for knowledge representation was proposed by Marvin Minsky in 1974, based on a special data structure to capture not only declarative but also procedural knowledge. In particular, the later was considered necessary to compensate for the limitations of logic-based declarative methods used so far, especially in relation to logical quantification, search control and combinatorial explosion [234].

The new data structure, called frame, was intended to provide a convenient way describe real world phenomena in a stereotypical form. These included aspects of objects, agents, actions, events and their different aggregations within a so-called frame-systems. For that purpose it also supported the embedding of explicitly procedures to resolve description items that are in conflict with observed facts or expectations.

It is important to point out that Minsky's work focused more on a system that could describe and reason about common-sense knowledge than in formal and logically precise knowledge. Indeed, the notion of frames as stereotypes stems from the need to embed systems with what can be considered *usually* the case given a situation, and not every logical possibility. Therefore a certain degree of bias and uncertainty was deemed not only acceptable, but also necessary in the process of commonsensical reasoning.

The premise behind Minsky's frameworks is that frames stand as idealized approximations of reality, hence subject to incremental refinement and improvement by a solver. For that purpose the need for higher levels of abstraction and meta-reasoning became evident, as a way to evaluate and guide the process of inference. This in turn required methods of storage and retrieval of relevant pieces of information, all described in terms of more general or specialized frames.

In this latter aspect, Frames provided a theoretical foundation for future research in Analogical and Case-based Reasoning, particularly in relation to learning, memory, and knowledge transfer between analog models. These concepts expanded the range of problems to be tackled by AI, leading to a particular interest on qualitative, common-sense reasoning using narrative-like knowledge structures.

Thus, different frames within a frame system were intended to enable different viewpoints of the same situation or narrative, without much concern for strong internal consistency. In this way, each frame can be considered as an unique collection of questions or issues typically associated to the overall situation described. Moreover, each frame should also provide indications for interpretation and resolution of conflicts.

In the example below a same ball-kicking event is being described from three different perspectives, depending on different questions or concerns (Table 6). For instance, the system may want to know either who kicked the ball, what was done to the ball, or what kind of event happened at all.

The reason why internal consistency was not seen as particularly relevant is that handling paradoxes is a fundamental trait of human intelligence, hence a desirable capability for a truly intelligent system. Part of that capability involves the provision of explanations

Table 6: Different sub-frame viewpoints for the same event frame. Adapted from [234].

Terminals	**Frame 1**	**Frame 2**	**Frame 3**
Subject:	The boy	The ball	Some ball-kicking
Action:	kicked	was kicked	happened
Object:	the ball	-	-
Time:	some time ago.	today.	this weekend.

for the paradox or impasses, alternative methods of resolution and excuses in case of failure. This in turn requires the representation of qualitative knowledge regarding spatial and temporal relationships, causality and side-effects, goals and constraints, among others.

At an even higher level, it also entails the notion of actors, their purposes and motivations, along with conventional pattern of behavior better described in terms of plans, narratives and stories.

4.1.4 Scripts

The ability to understand stories and other simple narratives is an example of this type of intelligence, which was further explored by Schank and Abelson with the development of the Scripts framework [274]. Scripts were in fact proposed as a cognitively inspired specialization of Frames. Whereas Frames provided stereotyped descriptions for a wide range of perceived phenomena, Scripts focused more on stereotyped human activities in particular social scenarios.

Scripts were defined as data structures describing an appropriate sequence of events within specific contexts, almost like a recipe. Similarly to Frames, the object-like structure of scripts consisted of slots and constraints over the values that could be used to fill them.

Stories were compositions of scripts, which in turn were composed by a series of stereotyped scenes. Each scene denoted a causal chain of actions appropriate for the context in which they take place. Scenes could also refer to other sub-scripts whenever more detailed information was required.

This hierarchical structure provided an efficient method for learning and memory, since knowledge of complex causal relationships could be stored in the form of reusable chunks or patterns [270].

Table 7: Different script viewpoints for the same script. Adapted from [274].

script:	restaurant		
roles:	customer, waitress, chef, cashier		
reason:	to eat food*		
scene 1:	entering		
scene 2:	ordering		
		ATRANS	receive menu
		MTRANS	read menu
		MBUILD	decide what self wants
		MTRANS	order to waitress
scene 3:	eating		
scene 4:	exiting		

Similarly to parallel efforts at the time, Scripts relied on production rules for inference of future events based on satisfied preconditions, thus following a forward search and bottom-up composition of actions. However, planning was also considered an important functionality, for which Scripts also provided an useful representation. In this case however, inference happened backwards in a top-down fashion, from high-level goals to the next level of action sequences. These in turn had their own set of sub-goals and action sequences at the lower level. This process of recursive decomposition continues only if needed, thus avoiding overload of information.

A variation of this approach was the computation of explanations about current states or facts based on the inference of past events, thus simulating abductive reasoning capabilities. Here, the characterization of relevant viewpoints proposed in Frames is explicitly encoded in the structure of Scripts by a list of roles.

Table 7 illustrates the example of a restaurant script from the point of customer, whose main goal is defined in the 'reason' slot. Other roles are included in the 'role' slot. Notice that scenes can be defined internally within the same script or invoked from other scripts. Thus, the generic scenes 'entering' or 'exiting' can be applied, with some adaptation, to different contexts with similar causal structure.

The Scripts framework, like similar computer models of the time combined deductive forward inference chaining, as well abductive backward chaining. The former was used for prediction of outcomes given known facts and state transformations, while the latter was used for hypothesis formation and explanation, as well as for automated planning, and eventually, functional modeling in design.

The next subsection introduces some of these aspects in the context of Hierarchical Task Networks (HTN), developed for automated planning.

4.1.5 Hierarchical Task Networks

An example of combined used of deductive forward chaining and abductive backward chaining can be found in automated planning techniques developed under the method of Hierarchical Task Networks (HTN). HTN planning is considered a generalization of classic automated planning methods like STRIPS, covering planning problems requiring partial ordering [266].

In HTN planning, a plan is formulated by repeated decomposition of high-level tasks into smaller sub-tasks, until a primitive level of tasks is reached. These primitive tasks are basic units of executable actions. What is new in HTN is the method by which plans get subdivided into smaller ones. The process is based, similarly to STRIPS, on the matching of preconditions of the high-level actions (or goal) to be decomposed with the effects of sub-plan patterns stored in a plan library [194]. The matching mechanism relies on constraints that specify the action or task. A HTN plan is thus a pattern, composed by several smaller stored patterns, usually specified by human experts as teleological models that are relevant within a domain [265].

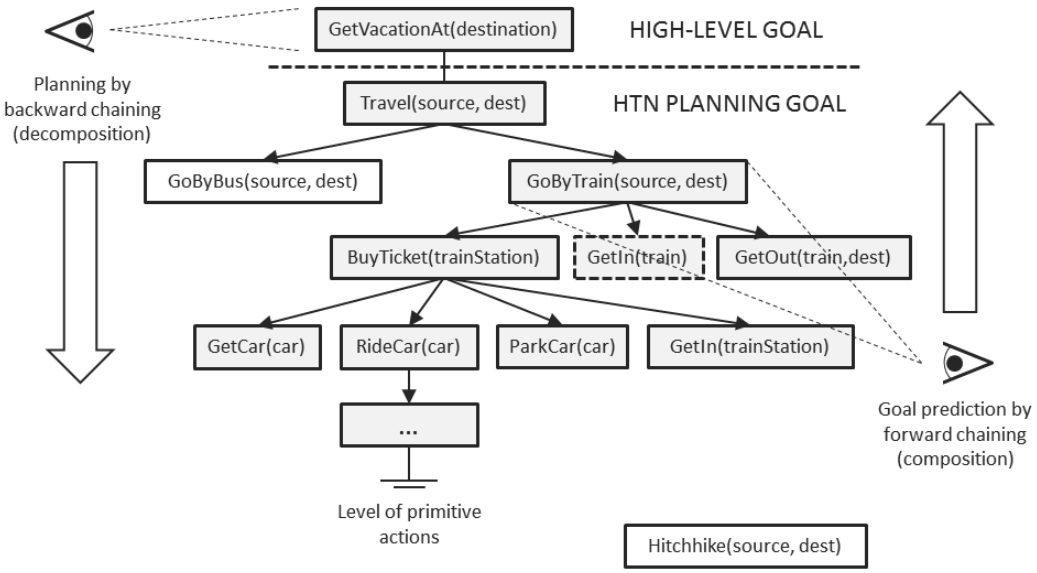

Figure 4.3: Two inference methods for the same hierarchical structure. Goal prediction by composition of plan and action patterns. HTN partial-order planning by goal decomposition into plan and action patterns. Stored sub-plans not fitting specified constraints are not considered (e.g. hitchhiking). Modified from [194]

.

Some of the issues of HTN in partially hierarchical domains is the selection of suitable plans based on user intent in relationship with effects and side effects of available actions. Complex combinations may arise from the interaction of both (e.g. when a side-effect becomes the intended goal of the user). To avoid this type problem, relevance clauses need to be introduced by making explicit the main purpose of effects, as usually understood within a domain. Thus, effects listed as *primary* would preclude spurious or non-minimal plans [194]. On the other hand, this approach may limit the range of solutions that could benefit from opportunistic use of side-effects, which by definition would no longer qualify as primary.

4.1.6 Analogical Reasoning

During the 1970s, the most widespread perspective on problem-solving was based on the early work of Newell and Simon discussed in section 4.1.1. In this perspective problem-solving involves search, selection and composition of pre-defined operators leading to a goal state. Different strands of means-ends analysis methods were developed as primary problem-solving approach. Within this theoretical framework, Freeman and Simon introduces the study of functional reasoning in design [136]. A different theoretical perspective was proposed by McDermott [233], Wilensky [325] and others, who equated problem-solving to planning. In this approach, knowledge is available to a solver in the form of pre-stored plans and not simply as primitive actions. Thus. each plan is composed by interdependent sets of sub-plans and actions, at different levels of abstraction. This allows the construction of a total solution in a recursive manner. Since a problem can have more than one plan available in memory, multiple solutions may exist for any given goal or sub-goal. Hence, part of the challenge is to provide not only the best solution, but also the rationale for the selection among many candidates.

In both perspectives of problem-solving, a major theoretical debate emerged regarding different approaches for knowledge representation and reasoning. In particular, there were competing views over declarative versus procedural representations, as well as logical versus heuristic reasoning methods. Eventually, certain level of consensus emerged towards the need for hybrid approaches involving multiple types of representation and reasoning [234, 293].

Yet, for more complex problems the combination of existing approaches was not good enough. Many aspects remained problematic and limited success was found in real-world applications. In this context, reasoning by analogy was proposed as a new, more general theory that could compensate for the previous limitations by accounting for key mechanisms of human cognition, such as memory and learning [141, 124]. Initial research on computational analogical reasoning stemmed from pioneer work of cognitive scientist Dedre Gentner and her structure-mapping theory [141]. The general goal of this work was to provide a framework to understand how people make sense of statements such “An electric battery is

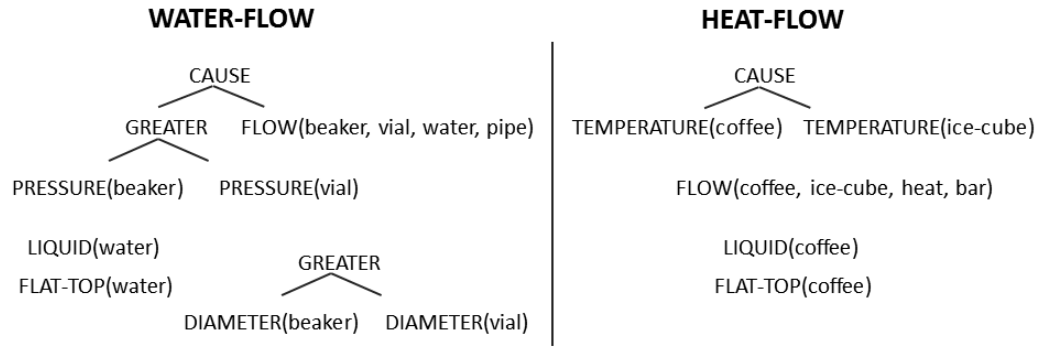

Figure 4.4: Simplified water-flow and heat-flow description for analogical mapping. From [124].

like a reservoir", or the "An atom is like a solar system."

According to this theory, one entity is analogically similar to another if it is possible to establish a mapping from the conceptual structure of the base entity to the conceptual structure of the target entity. In the examples above, both the battery and the atom are the targets, for which the analogy with the base entities referred to provides a plausible explanation.

In Gentner's theory, the analogical mapping is based only on the syntactic properties of the knowledge representation, independent from domain semantics. Specifically, it relied on the structure of relations, instead of attributes. The reliance on the relational structure of the representation was intended to differentiate analogies from other forms of comparison such as literal similarity and abstraction. Furthermore, it provided a flexible mechanism for drawing valid inferences with relative low domain knowledge of the new situation.

In the context of problem-solving, analogical reasoning offered an alternative method for situations where no specific heuristic rules or plans exist. In such cases, a new problem may be solved by analogical matching and transformation of old solutions of a similar problem pre-stored in memory [64].

The Structure-Mapping-Engine (SME) was the first computational model of Gentner's theory, showing early promising results in a number of applications and studies of human cognition [124].

However, the flexibility offered by context-free structural mapping came to a cost. The

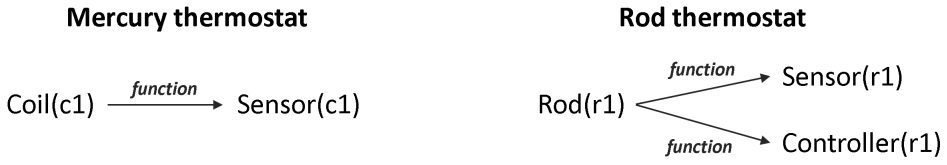

Figure 4.5: SME analogical mapping based on analysis of functional roles. From [121].

higher the number of, possibly arbitrary relationships to map, the more intensive the computation process became. Additionally, the lack of domain semantics to guide the process also increased significantly the risk of spurious results.

Carbonell provided an account on how of how previous knowledge about procedures and plans can be transferred by analogy to a new situation. In this case, the semantics of the domain clearly needs to be part of the knowledge representation. Moreover, the reasons why, when and how to apply certain plans and procedures also need to be included as a form of meta-explanation, i.e. justifications, for the solutions provided. The derivation or reconstructions of this meta level of information into new situations suggested a more powerful problem-solving mechanism, allowing a solver to evaluate and learn from past experiences [64].

The interest on practical applications with strong explanatory and justification capabilities led to the realization that both domain knowledge and general knowledge of situations are required to guide the process of analogical mapping. Among the efforts along this line was a reformulated version of structure-mapping called the Contextual Structure-Mapping (CSM), developed for abductive explanation of causality using analogical similarity.

Falkenhainer reports on the need for characterization of roles, including functional roles, as means to facilitate search and selection of relevant analogues [121, 122]. Figure 4.5 illustrates a many-to-one component mapping from base to target based on role analysis, where a mercury thermostat is found functionally analogous to a bi-metal rod thermostat.

Other early efforts in abductive explanation with characterization of functional roles were PROTOS [21]and SWALE [215]. In the area of automated planning the focus shifted towards the reusing the rationale for decision-making in previous planning episodes. Here,

analogical transfer took the form of derivations from decision traces found in the source analogue, to be used as control knowledge in the execution of different strategies of retrieval, adaptation, evaluation and learning [65, 311].

A key aspect to be considered in analogical reasoning is its utility and associated trade-offs in relation to the type of problem-solving task at hand, and more importantly, the novelty of the problem itself. This can be summarized in terms of a similarity spectrum, whereas in one end precedents or exemplars used as base entities of the analogical transfer are very similar to the problem at hand. In this case, the degree of novelty of the new problem is low, which facilitates efficient retrieval and adaptation of so-called "shallow analogies" [104]. However this has a narrow scope, being limited to problems that tend to recur often and are probably well known within single design domains, i.e. routine design.

On the other end of the spectrum, base and target entities are very different. This implies that the new problem is indeed very new, with no solutions available within its own target domain. This requires transfer of deeper analogical structures, possibly from different domains, involving higher degrees of generalization with multiple levels of abstraction.

While computationally demanding, it is this latter end of the spectrum which provides some of the most significant opportunities in design, especially during early conceptual stages. The next subsection discusses some of these opportunities, theoretical approaches and main challenges.

4.1.6.1 Analogical design

A comprehensive analysis of the role of analogy in design, and the challenges associated with the development of analogical reasoning applications to support design has been provided by Goel [153].

In this study, some characterizations of design found in literature have been provided, with attention to differences between routine, innovative and creative design. In order to systematize these differences, Goel proposes that creativity in design lies in a continuum, where variations in degree depend on two interdependent aspects of the design process.

These are the extent of reformulation in problem and solution, akin to the notion of co-evolution of problem and solution spaces discussed in chapter 1; and the degree of analogical transfer of knowledge from different sources to the new design problem [153].

The interdependency stems specifically from the choice of representation made to describe problem spaces. For instance, the Structure-Mapping model used in SME considers only the syntactic features of its representational structure. Any other analogy not covered by that particular syntax gets excluded from the mapping. In the context of design, failure to find a good source of analogy may be resolved by describing the problem in different terms, so to draw an useful analogy from a different source or domain. Eventually, good analogies may be found in the most unexpected locations.

In design, as in other complex process-driven problems, analogies can have different uses for different design tasks, stages or aspects of the whole design process. These uses are characterized by Goel according to four questions, namely, *why, what, how and when* [153].

The *why* refers to the type of task to be supported by the analogy, including but not limited to generation of design solutions. Other tasks along the entire design process may benefit as well. These may include:

- Interpretation of design problems
- Elaboration of design problems
- Decomposition of design problems
- Anticipation of potential difficulties with design candidates
- Refinement of design candidates
- Evaluation of design candidates
- Interpretation of evaluation results

The *what* refers to the content of knowledge being transferred to support these tasks. This knowledge may be about the domain itself or different types of control knowledge, i.e. strategic or tactic.

The question about *how* refers to issues of retrieval and knowledge transfer, according to the degree of similarity supported by the representation language.

The final question discussed in the study is *when*. This is related to strategic control over information processing, including when to generate, store and transfer new abstractions and other theoretical to issues of memory and learning.

In the context of this research the question about *how* is especially relevant. This is because it relates to the problem of providing a common representational language to support knowledge mappings across different domains. To better understand the problem, it is worthy to consider how the issue has been addressed in the past, especially in relation to the problem of similarity between analogues. From this perspective Case-based reasoning (CBR) is considered a limited case of analogical reasoning. This is because it operates in situation where there is a high degree of similarity between superficial features of new and old problems. Here the process of transfer is reduced to associative modification akin to variant or adaptive design [153].

Many of the earlier analogical design applications fall in the CBR category. The SEED system discussed in chapter 2 is an example. Other efforts also focusing on early conceptual design in architecture and engineering are Archie [251], Precedents [243], Design-MUSE [93], CADRE [183], and CadSyn and CaseCad [226].

In creative design however, deeper knowledge involving multiple abstractions and complex relational structures among objects and processes is required. In this context, various model-based representations have been proposed, with generic abstractions to describe structural, behavioral and functional dimensions of design at multiple levels of detail. In this category are FABEL [319], Dssua [258]and Ideal [22], for which Goel provides a comparative analysis in terms of the different types of generic abstractions adopted.

The generic nature of these abstractions are intended to satisfy the need for a high-level representational language able to describe commonalities across different domains. In this way deep analogical transfer of design knowledge across domains could be better supported. In this regard Goel provides an important operational definition, also from [153]:

> "Analogical design involves learning and transfer of generic design abstractions from one design situation to another, where the generic design abstractions specify the structure of relations among the elements of a design problem, solution, domain, or strategy, and where the transfer can occur in the service of any design task in the new situation."

This definition of analogical design extends the scope of application beyond design generation or refinement of design alternatives, providing a theoretical framework to support a wider range of tasks on which the design process rely. This include the possibility of providing explanations and justifications relevant for each task situation, as pertaining to structural, behavioral or functional aspects of design.

The key aspect however is the provision of such generic abstractions, in representational language that is compatible to the domain-specific languages used in the analogy source and target. Furthermore, in order to support a wide range of design scenarios such abstractions need to convey rich relational structures in the form of design patterns reusable across domains.

There are two possible approaches for the construction of these design patterns. They could either be generated automatically by intelligent systems using inductive inference [121, 311] or by differential diagnosis and model-based simulation [22, 155]. Another option is that generic design patterns could be modeled by domain experts, in order to be used by systems in support of design tasks. In most cases, it seems that the key for inductive generalization of design pattern lies on the formulation of models of function and behavior with multiple levels of abstraction.

Indeed, this observation stems from common sense, since people give names to many objects precisely as an abstraction for the behaviors they tend to perform under common circumstances. When some of these behaviors become more relevant than others, they are called *functions*. However, analogies may however operate on any behavior implied by the object's name, besides nominal functions.

4.2 Functional representation and functional models

In a general sense, all AI models described so far rely on some notion of function and behavior, with various ad-hoc approaches to represent them. It was not until late eighties and nineties, with the growing adoption of CAD systems in various design industries that the interest in formalizing these notion became widespread.

Efforts from within the design community, both in academia and software industry were taking place at the same time, but focusing mostly on the informal description of behavioral properties and functional requirements. These were represented either as simple text or as constrained values over design properties. This approach facilitated database implementation and interoperability with emerging parametric CAD technologies. Some of these early efforts as they relate to the AEC industry were already reviewed on chapter 2. Despite its obvious limitations it remains to a large extent, the predominant paradigm today, as discussed in chapter 3.

Yet, the potential advantages of a more formal approach became apparent, especially given the favorable prospects of research in analogical reasoning. These prospects led to the formulation of a number of different functional representation schemes. Unfortunately, the goal of providing an operational machine-readable representation of design functionality remained elusive.

Part of the challenge stemmed from theoretical issues of computational representation and algorithms. Others are fundamentally philosophical, pertaining to the multi-faceted nature of design, and design functionality. The latter gets reflected by the diversity of interpretations given to the concept of function found in literature [167, 85, 73]. Clearly, each interpretation is tailored to fit domain-specific world-views, reflecting disciplinary assumptions, methods and goals.

Vermaas proposes that the 'elusiveness' of the notion of function, and therefore, the lack of consensus about an unifying general meaning, stems not so much from its ambiguity, but from the resistance to accept it as essentially ambiguous [316]. This suggests that the search for precision and consistency in the definition of function might be misguided, as it prevents not only desired levels of generality, but also expressivity required to support compatible

interpretations across domains.

Given that design problems are becoming increasingly more interdisciplinary, with a demand for design applications that are more interconnected and interoperable, the need for a general, expressive and robust representation of functionality is also growing. Moreover, recent developments in bioengineering, ecological engineering and biomimicry are also requiring semantic unification able to cover biological functions as well.

Furthermore, Crilly makes the case for the expanding the semantic coverage beyond technical functions, in order to include other functional dimensions of design [85]. Architectural design is certainly a relevant example, where a number of different functional meanings exist that require reconciliation, ranging from technical and biological to psychological, social and organizational functions.

Current debate over different meanings and uses for the notion of function touch on a wide range of issues to be taken into consideration. The possibility of a comprehensive synthesis however requires a systematic analysis of previous efforts, in order to find general features within the context of domain-specific requirements. But such an effort is beyond the immediate scope of this research. Instead, a rather selective overview is provided in the following subsections. These were chosen because they provide theoretical principles that are general enough to cover the most relevant functional aspects of architecture and Building Design.

4.2.1 The Structure-Behavior-Function schema

The Structure-Behavior-Function (SBF) schema was developed by Goel et al. as a model-based representation of functionality based on explicit description of causal processes enabled by the structural composition of an artifact [152, 156]. The SBF schema has its roots in a series of previous research efforts to address the representation of functionality of devices. In particular, SBF evolved from the integration of two previous functional representation models [256]. These were an earlier version of the Functional Representation (FR) schema by Sembugamoorthy and Chandrasekaran [278, 159, 74] and the Method of Consolidation of Bylander and Chandrasekaran [62].

One of the main contributions of SBF was the unification of the problem-solving perspective on design with cognitive aspects of memory and learning in a single integrated framework [159]. This approach was grounded on the work of dynamic memory [271, 272], and the role of retrieval and organizational strategies in conceptual memory [207], leading to the new paradigm of case-based reasoning [262, 206].

Kritik was an early system for conceptual adaptive design implementing model-based and case-based reasoning [159]. Since its focus was on cognitive aspects of memory and learning, Kritik was developed to support two different types of mapping. The first one from function to structure as a method to guide case retrieval and modification. The second is the inverse mapping from structure to function, as method for qualitative explanation of working principles, diagnosis and repair [150].

4.2.1.1 Functions, side-effects and side-functions

Similarly to FR, functions in SBF were originally defined as intended subset of all output behaviors performed by an artifact. Since a design task was understood as a function-to-structure mapping, a functional specification could be used as index for searching of relevant cases. A candidate case in turn was represented as a SBF model describing the function of the device, its structure, and the set of internal causal behaviors that make it work.

The introduction of causal behaviors represented a new ontological commitment, allowing the original function-to-structure mapping to be articulated by an intermediate layers of abstraction. This added layer of causality led to the dual meaning of function in SBF, both as intended output behavior as well as behavioral abstraction. That is, a reference for the causal links within nested chains of functional dependency.

In a SBF, a function is specified as a schema, with slots specifying a set of preconditions (i.e. GIVEN), post-conditions (i.e. MAKES), and a pointer to the main behavior that accomplishes the function (i.e. BY-BEHAVIOR).

Thus, the BY-BEHAVIOR pointer connects a function to the level of abstraction immediately below, which provides a high level explanation of its behavioral mechanisms. The function schema also include the contextual conditions under which the referenced behavior

achieves the function. These include structural configuration and a set of external stimuli. A behavior is represented as a sequence of states and transitions between the states, each represented by a specific sub-schema.

It is important to point out that the characterization of function in SBF remained consistent with the one originally formulated in the FR schema, in terms of meaning both an intended subset of output behaviors as well as an abstraction of internal causal behaviors. An important distinction however is that SBF allowed anticipated behavioral *side-effects* to be described explicitly in the schema [159, 158]. Figure 4.6 depicts the function "Change Angular Momentum of Telescope". The schema includes "Generation of Heat in Bearing" as anticipated side-effect.

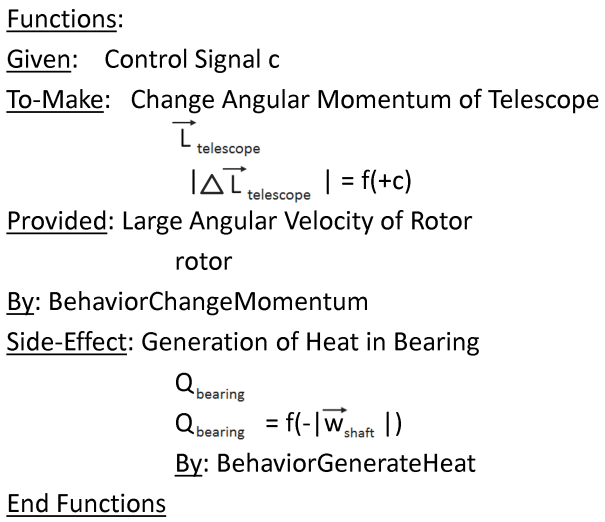

Figure 4.6: Function schema. Given preconditions, To-Make post-conditions, By pointer to output behavior and anticipated Side-Effect. From [159].

The inclusion of side-effects served two purposes. First, it allowed the evaluation of alternatives during retrieval in cases where side-effects where in violation of additional functional constraints. Secondly, side-effects could be support the accomplishment of additional functions. In Kritik, these additional functions derived from useful side-effects were called *side-functions* of the primary function. Functions can also have *sub-functions* as part of them, following the traditional functional decomposition method. Finally, *supporting functions* were also included as part of Kritik's taxonomy, as behavioral pre-conditions to

enable sub-functions. Altogether they allow a high-level representation of causality in the form of a directed acyclic graph [159].

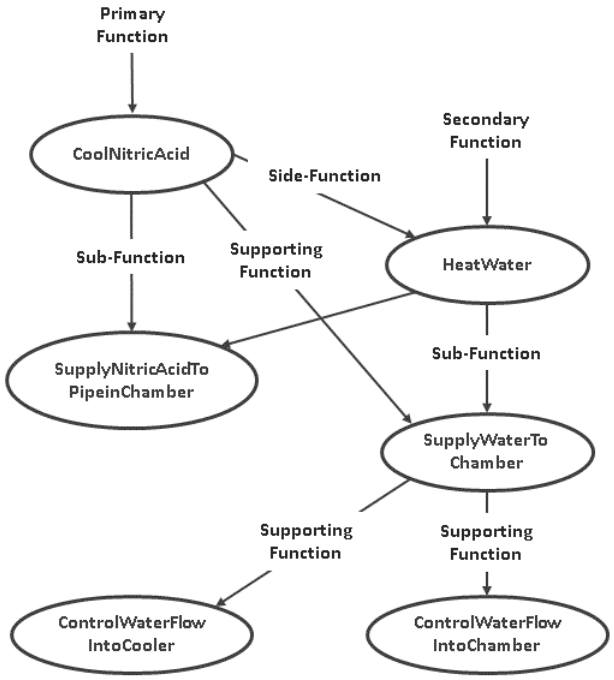

Figure 4.7: Semi-lattice of primary function, side-function, sub-function and supporting function. Adapted from [159].

4.2.1.2 Generic design patterns

During the 1990s, the range of similarity covered by SBF was expanded, in order to tackle domain-independent analogical design problems. IDeAL was a computer program developed to that purpose, based on the concept of domain-independent design patterns. This essentially involved the abstraction of known design solutions described as SBF models into generic behavior-function (BF) models - the design patterns - that could be retrieved and adapted to solve new problems [154].

The concept of design patterns was originally proposed by Christopher Alexander in the field of Architectural Design [9], but had a significant influence in the areas of computer science and software engineering [139]. In these areas, design patterns are implicitly associated with productive units of reusable design knowledge. In IDeAL however, design

patterns were explicitly used as productive units of analogical transfer, involving not only issues of automated reuse, but also retrieval, evaluation and learning [155].

A design pattern can be also understood as a specification of generic teleological relations holding among abstract design elements. In IDeAL, these patterns were called in Generic Teleological Mechanisms (GTM), and represented as behavior-function models. The concepts of feedback in control systems and cascading are examples of such GTMs [22].

4.2.1.3 Environmentally bounded functions

Eventually SBF evolved into a family of representation models, addressing different design problems and aspects of the design process (i.e. tasks). These included functional modeling for design adaptation of devices based on interactions with different operational environments[256], understanding the functional meaning of design drawings using compositional analogy [330] and cross-domain analogical transfer for Biological Inspired Design (BID) [310, 172].

In the case of adaptative design of devices for new environments, an extension of SBF was proposed, called Environmentally-bound SBF (ESBF). This extension was envisioned to support modeling of environmental interactions, and the functional role they play in the overall functionality of devices [256].

Interestingly, the notion of behavioral side-effect and side-function introduced in the original SBF schema do not seem to play a relevant part in the characterization of such functional roles. Rather, the environment is treated implicitly as extension of the boundary conditions of the device, supporting only a predetermined sets of interactions and constraints.

4.2.2 The Functional Representation (FR) schema

The Functional Representation (FR) scheme initially developed in the 1980s by Chandrasekaran and team as an effort to provide a machine-readable description of *device-centric* functions. The main goal was to support automation in the diagnosis and repair tasks, by providing explanations of causal processes responsible for the overall functionality of physical devices [278].

Later, the FR scheme was applied to cover other aspects of the design process under a compositional modeling approach [123]. These involved issues of selection and retrieval of components from component libraries using functions as indexes, as well as verification of design functionality based on the description of structures and causal behaviors associated to each retrieved component.

However, this method is only appropriate in design domains with well-defined problems, and for which the kit-of-parts approach to design is suitable. For complex, ill-defined design problems involving multiple disciplines a more general representational framework is required. For that purpose, a revised FR scheme was proposed by Chandrasekaran and Josephson, by introducing a simple ontology intended to cover different interpretations of the terms 'function' and 'behavior', as normally found in engineering practice [76].

This revised version of FR waa an attempt to provide a minimalist, yet expressive ontological framework, that could unify not only meanings, but also modeling requirements of different disciplinary domains. Generally, these involve other semantic complexities, which can be categorized as dichotomies or extremes on a spectrum. According to Chandrasekaran and Josephson [75], these include:

- *Dynamic versus static functions.* The first indicates functions that are satisfied by certain property changes overtime, normally represented as state-change behavior. The second indicates functions satisfied by virtue of the structure of the object itself. For instance, a door lintel, a paperweight.

- *Functions of objects versus functions of processes.* It is normally assumed that both objects and processes have functions, since they are designed or planned towards some goal. This view is ontologically problematic and will be discussed in the next chapter.

- *Functions as states versus function as processes.* This dichotomy depends on the level of granularity in which a phenomena is represented. Thus, as state represents an abstract view of underlying processes. Conversely, a process can be represented as a sequence of state changes.

- *Functions involving intentional agents versus functions of subsystems in larger systems.* This dichotomy is later resolved in FR with the distinction between *device-centric functions* and *environment-centric functions.*

- *Functions involving creation and destruction of abstract or concrete entities.* Related to the function of creating or destroying entities, such as the function of the immune system in killing bacteria, or the function of computers in creating information artifacts.

Particularly relevant for the research of this dissertation is the distinction between *device-centric functions* and *environment-centric functions*, and the types of causal interactions that occur in-between.

In order to account for all these distinctions in a compact but general manner, the FR ontology is grounded on the idea of function as *effect.* From this, a the small set of concepts follows. These include *behavioral constraint*, *role*, *functional role*, *mode of deployment* and *intentionality* of agents. The next subsections discuss these notions, following a similar order to the work of Chandrasekaran and Josephson presented in [76]. All references are from this source, unless indicated.

4.2.2.1 Causal processes

The basic semantic unit of the FR ontology is an *object.* Objects can stand in structural and causal relationship with other objects according to specific *views.* For example, a pocket radio can be represented from viewpoint of being used as a radio, or from the viewpoint of being used as a paperweight.

The representation of objects include the description of causal variables, internal causal relations among them and terminals specifying the interactions with other objects and the external world. These include stimuli or input actions that objects admit, as well as the set of outputs that objects produce as result of their internal causal behaviors.

In FR objects can be related by means of different causal interactions. These can be represented as parametric dependencies between terminals and variables, or by actions

leading to deeper structural changes. For instance, objects being moved, created, destroyed or changing modes.

Further, a collection of objects standing in causal relation may itself be abstracted as an object. This shift allows representation and reasoning at different levels of granularity. Notice however that such relation does not entail necessarily structural composition or connectivity in a way intended by the design (e.g. physical assemblies). Hence, more abstract object aggregations are also permitted, insofar the nature of their behavioral interactions are relevant from some point of view.

4.2.2.2 Multiple meanings of behavior

Different abstractions pertaining to different views can be related through mappings rules. This allows moving across different levels and viewpoints, in order to support reasoning about different phenomena of interest.

At a general level, a function according to FR is a subset of output behaviors exhibited by a device. In this way, it shares the same interpretation than other schemes, such as SBF. However, a further characterization of the meaning of the term 'behavior' is required. In this regard, five possible meanings are identified as relevant, which are characterized in terms of the primitive notion *state variable*:

- *Interpretation 1:* Behavior as the value(s) or the relation between values of state variables pertaining to an object of interest at a particular instant. Example: 'How did the car behave? It rattled when I hit the curve.'

- *Interpretation 2:* Behavior as the value(s) or relations between values of the properties of an object. Example: 'A window *transmits* light from outside to inside the house.'

- *Interpretation 3:* Behavior as the value(s) of state variables of interest over an interval of time. Example: 'What did you notice about the behavior? The horse power had a momentary increase, then it started to decrease.'

- *Interpretation 4:* Behavior as the value(s) of state variables specifically labelled as 'output' state variables, either at an instant or over an interval. Example: 'The

amplifier is behaving well; its output voltage is constant.'

- *Interpretation 5:* Behavior as the values of all state variables in the object description, either at an instant or over an interval of time. Example: 'A graph plotting all the variables over time.'

The idea that behavior is synonymous or that it can be characterized as values of state properties is problematic without consideration of other aspects of the physical phenomena being referred to. In isolation, these characterizations lack the semantics required for more advanced forms of reasoning, as discussed in 3.4.3.

4.2.2.3 Function as effect

Since functions are considered subsets of output behaviors, different meanings of behavior entail different meanings of function. It is for this reason, partially at least, that providing a single "true" definition of function has been found to be extremely difficult, if not impossible.

Hence, according to the FR scheme, a range of possible meanings for function should be not only admissible, but necessary. For that purpose, the authors indicate what is common among all meanings of function in engineering practice, namely, the idea that an object, artifact or actor are either *doing* something or have properties that are *intended* or *desired* by someone [76].

Here, the difference with similar proposals in functional representation research is the explicit introduction of intentionality. By making this assumption explicit, all the other notions, such as structure, behavior and causal models need to be considered neutral regarding purpose.

This is an important distinction because it accounts for the *effects* that objects, artifacts and actors may cause, independently from what is assumed to be intrinsic purposes. With this distinction, functions can be characterized explicitly as *intended* or *desired* effect, while at the same time other effects may exist that are not intentional.

Effects can be described according to different viewpoints or levels of abstraction. A *device-centric* viewpoint describes the internal causal processes responsible for the effect,

without reference to the external world of the device. On the other hand, an *environment-centric* viewpoint describes the interactions between a device and its environment leading to the effect. Often, descriptions may take a mixed viewpoint.

In general, environment-centric descriptions provide a more abstract point of view relevant to the achievement of high-level goals or needs. For this reason, it is convenient to refer to *device-centric functions* and *environment-centric function*, as each imply different types of intended effects, for which more specialized descriptions may be required.

4.2.2.4 Function as behavioral constraint

Another common aspect in all meanings of function identified by Chandrasekaran and Josephson is the characterization of function in terms of *behavioral constraints*. A behavioral constraint specifies the properties of behaviors intended to be satisfied under certain conditions, as exemplified in the five meanings of behavior mentioned previously. Despite some of the problems of equating behavior with constraints over behavioral properties discussed before, in FR this idea gets further developed to allow the level of generalization intended.

In FR, behavioral constraints are expressed as predicates of the form `P(B)`. Thus, an object is said to satisfy a behavioral constraint `F` if the values of relevant variables in the object satisfy the predicates under the conditions specified in `F`.

A typical specification of a behavioral constraint takes the form: `If C, then P(B)`. For example: if switch is pressed, then the voltage at the output terminal increases to certain amount. This conditional form can also be composed into larger sequences of rules, such as `If` C_1`, then` P_1`(B); then If` C_2`, then` P_2`(B),... If` C_m`, then` P_m`(B).`

4.2.2.5 Mode of deployment

Use case scenarios involving multiple interactions, such as those required for ATM or vending machines, often rely on the specification of behavioral constraints. However, from an environment-centric point of view, further specification regarding contexts of use is also needed.

In FR, this specification is called *mode of deployment*. Thus, a mode of deployment

describes external conditions required so that a device can deliver its intended effects, i.e. its function. These include the conditions of use or operation under which relevant causal interactions apply.

The intuition is represented if FR by the relation `M(W,D)`, where `M` is the mode of deployment, `D` is the device and `W` is some world where the device is being deployed.

More specifically, the relation `M(W,D)` specifies how the set of causal interactions between `D` and entities in `W` are expected to occur. Such specification may include structural relations between `D` and entities in `W`, as well as sequences of actions to be taken by entities in `W`.

Typically, the specification of relations with the environment requires the specification of terminals or interface variables. Moreover, it also requires the specification of internal and external mechanisms capable of reacting or processing such interactions (e.g. feedback). Although this last requirement is relatively straightforward for a small number of intended effects, it imposes major modeling challenges when the number of interactions increases, given the risk of negative effects.

Ultimately, the relevance of effects, and therefore the need for control or mitigation, depends to a large extent on the goals and priorities associated to different stakeholders involved.

4.2.2.6 Roles and functional roles

To provide a general framework in which various meanings of function could be unified, the notions of *role* and *functional role* are introduced. The key distinction lies on the criteria of intentionality. Thus, a *role* is characterized as follows:

> If an object `D` is introduced in some world `W` according to a mode of deployment `M(D,W)`, and object `D` happens to cause the satisfaction of the set of behavioral constraints `F` specified for `W`, then it is said that `D` plays, or performs the *"role"* `F` in `D`.

Notice that in this case, roles can be performed by objects irrespective of the intentions of an agent. Chandrasekaran and Josephson point out that 'role' is just a descriptive term, with no bearing on intentionality. As such, it serves to capture an aspect of the effect caused

by objects on their environment, for which no need or desire have to exist. A *functional role* on the other hand is characterized as follows:

> If an agent `A` desires or intends a set of behavioral constraints `F` to be satisfied in some world `W`, in a mode of deployment `M(W,D)`, and object `D` is introduced into `W`, so that `D` happens to cause the satisfaction of `F`, it is said that `D` has, or performs the *functional role* `F` in `W`.

A ledge along a hiking trail is given as example of the relationship between roles and functions. In particular, it helps to illustrate the fact that functions exist even before a structure is created or invoked to produce the intended effect. Thus, if a person finds a ledge to sit on, then the ledge not only performs the *role* of a chair, but it also performs the *function* of a chair.

The characterization of function as role is relevant for the purpose of providing a general framework able to unify different domain-specific interpretations of function. The expectation of generality is based on the possibility of articulation between intention-neutral and intention-intensive interpretations of 'role", as found either in scientific or engineering discourses. Under this criteria, the characterization of function as a role is showed to apply to at least three difference scenarios:

1. Artifacts (e.g. devices and systems) explicitly designed for a function.
2. Artifacts and other entities that just happen to be used by someone for a certain function (e.g. a ledge used as chair).
3. Biological entities that fulfill a function.

The notion of 'role' make these characterizations compatible with each other, by denoting causal *effects* as common denominator. Intentionality on the other hand is a *qualification* ascribed to a role in situations where the effect is beneficial, necessary or desired by someone. In the case of biological entities, intentionality is assumed to be intrinsic, perhaps as result of evolutionary demands.

From these observations, Chandrasekaran and Josephson offer the following, informal definition: *Functions are Roles + Intentions*

It follows that objects can be assigned multiple functions, each involving a specific role, intention and mode of deployment. Scenario number two in the list above illustrates this point. In particular, it indicates the opportunistic and often idiosyncratic ways by which people may take advantage of the properties of an object, artifact or otherwise, given certain circumstances.

This is particularly relevant in design, where both conventions and possible idiosyncrasies need to be taken into consideration through anticipation of different modes of deployment. Yet, it also indicates that intentionality over roles is not always a prerogative of the designer.

4.2.2.7 Relationships between device-centric and environment-centric functions

The primary purpose for having the distinction between device-centric and environment-centric functions is to support inferences about the conditions of satisfiability for high-level goals under some use case scenario.

According to FR, this involves a process of transformation from goals or needs into more specific model-based descriptions of environment-centric functions. Only then it is possible to assess if an artifact can play a functional role, given its ability to deliver the intended environmental effects under specified conditions.

For example, given the functional role $\mathtt{F_E}$, in some environment `E`, the design problem is to find or create a device `D`, with device-centric function $\mathtt{F_D}$, so that $\mathtt{F_E}$ can be inferred from the models of `D`, $\mathtt{F_D}$, `E`, and mode of deployment `M(D,E)`. If $\mathtt{F_E}$ is obtained, then it can be said that `D` performs the functional role $\mathtt{F_E}$ in `E` under `M`.

This characterization of function as role however is general enough as to cover other aspects of design teleology besides primary roles intended *a priori* , i.e. primary functions. For instance, when designers take advantage of known side-effects to support a secondary function, either internal to the artifact or external. In this case, the component causing the beneficial side-effect acquires an additional secondary functional role, or side-function [159].

However, there are situations where unintended roles may also occur, often *a posteriori* and in an unanticipated fashion. In these situations, at least two scenarios are possible:

- Objects perform unintended roles with negative side-effects.
- Objects perform unintended roles with positive side-effects.

Both scenarios are relevant in design verification, especially regarding quantification of performance, as well as in diagnosis and explanation. Unfortunately, the possibility of unintended roles with negative side-effects is usually not captured in design models.

On the other hand, if mechanisms for control and mitigation of negative side-effects are adopted, the rationale behind their adoption is usually not captured in models either. Without the rationale, these control mechanisms can be undone or jeopardized during design, operation or repair.

While the implications are certainly problematic in a device-centric functions, they are even more so in an environment-centric ones, where behavioral interactions among heterogeneous entities tend to compound in unexpected ways.

To deal with the problem it is necessary to include unintended roles as part of the model-based description of the entire system, but without pre-qualification of effects in terms of positive or negative impact. Such addition would improve the completeness of functional models, facilitating the development of specialized abstractions to describe different *aspects* of effects, according to domain-specific viewpoints. This in turn would facilitate the specification of mappings required for coordination and conflict resolution among different functional requirements.

Certainly, only relative completeness can be pursued, given the known relevance of certain roles and their effects for certain types of systems [76]. Still, uncertainty arises because incomplete role information. This may be due to the lack of details about the role of components or component combinations, or because there are other factors (e.g. objects or agents) performing a role that are unknown [121].

4.3 Discussion on functional representation and reasoning

The review on the evolution of teleological models and functional representation in Artificial Intelligence is motivated, similarly to the review on the evolution of building models, by the need to identify predominant theories and models, as well as strengths and weaknesses of different approaches. More specifically, the goal is to identify common trends and patterns that may provide a direction in the development of an operational functional modeling framework for Building Design.

4.3.1 Heuristics, stereotypes and levels of abstraction

As it has been reviewed in Chapter II 2, initial attempts in functional representation in Building Design applications were very much influenced by the problem-solving paradigm introduced by Newell and Simon with the GPS program [239, 238]. Given initial promising results in the so-called 'toy worlds', researchers started to explore the applicability of this new model in Building Design [101, 107] and planning [127], using of a mix of declarative, logic-based reasoning approaches, as well as heuristic methods.

Soon however, a number of theoretical limitations became evident, especially in relation to the treatment of design goals and functions in terms of constraint formulations [136]. In order to provide a more general and expressive framework, a number of frame-based schemes were introduced, with a mix of declarative, logic-based and procedural methods for searching solution spaces. Strategies for knowledge organization were explored by composition of reusable chunks of design meaning, i.e. semantic patterns. This allowed the encoding of design knowledge at higher levels of abstraction in an operational manner. An important theoretical contribution in this regard was the frame model proposed by Minsky. Here, the intent was to provide a more expressive and flexible form of knowledge representation capable to deal with stereotypical situations typically faced by people on a daily basis, and for which common-sense reasoning was often the most convenient problem-solving approach. This involved the development of different control strategies at *multiple levels of abstraction*, as well as inference capabilities under conditions of partial or contradictory knowledge. Within the context of goal-oriented systems, these challenges involved reasoning according

to generic principles of causality, including side-effects [234].

4.3.2 Qualitative reasoning: partial knowledge, explanation and rationale

Story understanding and automated process planning were two areas for which knowledge-intensive applications based on qualitative reasoning were developed. These imposed new theoretical challenges regarding knowledge representation, organization and reusability. New computer models were proposed, in which the basic units of reasoning were no longer primitives, but compositions denoting complex, *qualitative behavioral patterns*. These patterns represented knowledge about stereotypical situations, described in terms of plans, narratives and stories [274, 266]. These 'chunks' of knowledge reduced the computational cost of having to build solution from first-principles, while at the same time facilitated the implementation of different strategies for knowledge representation, including memory, retrieval and processing. Impasses at one level of abstraction were handled at higher levels with access to domain-specific knowledge about situations, goals, actors, as well as procedures for evaluation and conflict resolution.

The ability to derive *explanations* for a story or the *rationale* behind the decisions of a planning model became an important area of research, especially in the context of real-world applications, were confidence in automated solutions needs to be justified by evidence of sound reasoning [64]. However, logic-based or heuristic methods did not provide enough reasoning power and flexibility to deal with novelty and complexity of many design tasks, especially under conditions of incomplete knowledge.

The problem of how to draw *plausible inferences* with *partial knowledge* started to generate growing interest in a number of different fields. Interdisciplinary research efforts between Cognitive Science and AI in the late 1980s postulated analogical reasoning as one of the key drivers behind such capability. The collaboration led to the development of a number of computational models attempting to simulate different aspects of *analogical reasoning* in humans, including mechanisms for memory indexing, similarity matching and knowledge transfer by means of structural mapping [141, 124].

4.3.3 Analogical reasoning in design: roles and context

Initial findings motivated the application of analogical reasoning principles in the new research area of Analogical Design [153]. The basic idea was to enable the retrieval and reuse of known solutions from previous design problems in the resolution of new design problems, based on some criteria of similarity. Thus, the potential for reusability is dependent on the criteria adopted for assessing similarity within a spectrum. A high degree of similarity, especially between structural features, typically takes place within single design domains, whereas similarity of causal mechanisms and functional principles can take place across different domains. Case-based Design (CBD) represents a limited version of Analogical Reasoning concerned with the former scenario, where the level of abstraction required in representation and reasoning is generally low [153]. This was considered a potentially useful approach in domains lacking formal theories and for which reliance on precedents is strong. Among these domains were architecture and engineering design, where a significant number of valuable research efforts were developed. In particular, there was a special interest on supporting conceptual stages in scenarios involving adaptative design [152, 251, 258, 243, 93, 319, 183, 226, 131]. However, an important limitation of CBD was the propensity of the process of similarity matching and knowledge transfer to be driven by superficial design features, eventually leading to 'shallow analogies' with little relevance to the problem at hand.

Deeper analogies in turn required richer abstractions with more meaningful sets of relationships. This is necessary not only to generate novel solutions but also to justify the rationale driving the analogical reasoning process. In this regard, the identification of causal *roles* played by different entities in the *base* of the analogy provides an important criteria to guide similarity mapping and knowledge transfer to the target, i.e. the new design.

In the context of physical systems, the role of an entity is based on how it contributes to the system's overall behavior, whether the contribution is intended by someone outside the system, or not. The former corresponds to a functional role, denoting how an entity of the system is intended to satisfy a particular goal or purpose [121]. Yet, uncertainty arises due to incompleteness of knowledge about the existence of other relevant entities in the system,

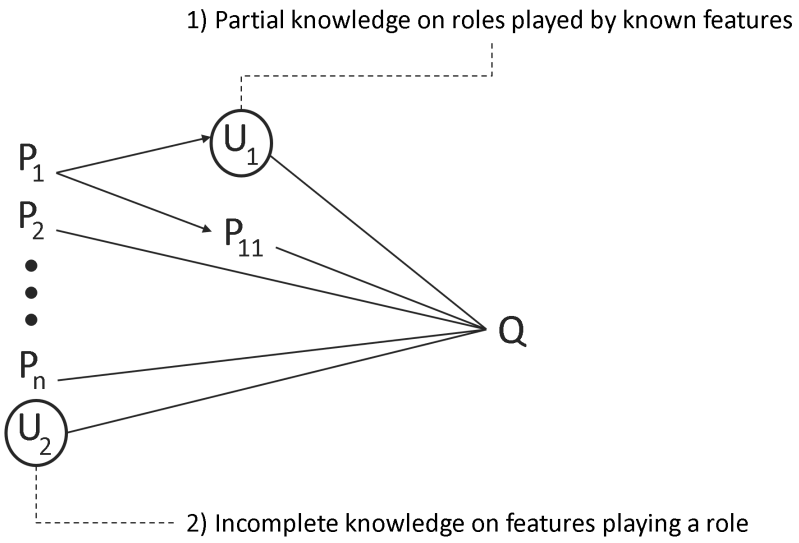

Figure 4.8: Role uncertainty under Open World Assumption. Uncertainty due to 1) partial knowledge of feature's roles; and 2) incomplete knowledge of all relevant features. Adapted from [121].

or about additional roles that known entities may play in the context of analogues [122]. Diagram in figure 4.8 illustrates these two forms of uncertainty.

4.3.4 Functional modeling in engineering design

Meanwhile, parallel efforts in theory and methods of engineering design were attempting to provide a more systematic characterization of functions and functional requirements. These efforts were motivated by the need for continuous innovation in product development and manufacturing, within increasingly faster time-to-market production cycles at lower costs. To that end, interdisciplinary collaboration and automation of various design tasks were considered critical pre-requisites. This situation led to a growing interest for integrated design environments and computer applications able to support various stages of the product development life-cycle, ranging from requirements engineering and conceptual design, down to various forms of design optimization, especially in relation to manufacturing. In this context, the representation different functional aspects of a product became critical, demanding a robust, yet flexible modeling framework. Principles of compositional modeling suggested the possibility of representing functional knowledge at different levels of abstraction by means of functional decomposition, while modularization would facilitate knowledge

reuse and extensibility.

An important issue to be addressed however relates to the 'conceptualization' of the domain of application, and how it supports different notions of function, behavior and structure according to different product life-cycle perspectives. In this regard, different research groups in the engineering design and AI communities have proposed a number of representation schemes, each characterized by a series of particular ontological commitments. These in general tend to be strongly influenced by the immediate purpose and scope of target applications, and the predominant theoretical paradigm of each domain. Thus, proposed functional representation schemes vary considerably in degree of generality, reflecting different conceptual approaches regarding definition of entity and relationship types, as well as strategies for their representation.

The work of Kleer and Brown [90], Gero [142], Keuneke [198], Umeda and Tomiyama [308], Stone and Woo [295], Kitamura and Mizoguchi [200], Chakrabarti et al. [72], as well as the work of Chandrasekaran' FR and Goel's SBF, are among the most important examples. Erden et al.[118]provides a comparative analysis of these schemes in relation to some of the most popular models and methodologies proposed in design theory and product development. These include the systematic approach to engineering design, by Pahl and Beitz [247], the axiomatic design principles, by Suh [297, 298], and failure mode and effect analysis (FMEA) [260]. However, despite attempts to provide a comprehensive framework, the level of generality and formalization required for the representation of different functional aspects of design remains limited.

Part of the challenge has to deal with the level abstraction chosen for the description of functionality, in relation to different strategies of structural composition. Thus, some schemes are predominantly device-centric, focusing on the description of artifacts and their internal causal relations, while others prioritize interactions at the system level between devices and their operational environment. It is in this distinction that the characterization of function as *role* initially suggested by Falkenhainer [121, 122] becomes critical.

In particular, the shift towards an environment-centric perspective requires the consideration of all relevant roles that different system components may play, including those

resulting from behavioral side-effects, which may have an impact on the satisfaction of high-level, global needs. While SBF does admit the representation of all output behaviors of a device, including anticipated side-effects at various levels of abstraction [159, 151], these are described according to their nominal roles within internally defined composition hierarchies, and under clearly delimited boundaries and interfaces. This is consistent with the interpretation of function in SBF as behavioral abstraction, insofar the behavior referred to is internal to the device. However, this poses the problem of how boundaries are defined when taking in consideration environment interactions that do not conform to a single, predetermined hierarchy.

In fact, there is a degree of subjectivity behind the delimitation of systems' boundaries, which imposes the need for additional levels of flexibility when it comes to the representation of functionality from a more global perspective. This problem is preliminarily addressed by Chandrasekaran and Josephson in their discussion about environment-centric description of functions. According to these authors, humans tend to apply such boundaries by following an heuristic approach, in which a device is assigned a function corresponding to the lowest level of abstraction in the chain of needs the device can satisfy (in [76], pg. 174).

Thus, an ATM is normally assigned the function of allowing cash to be withdrawn, but not the function of allowing the purchase of goods. The latter corresponds to a higher level need that may indeed be satisfied by the withdraw of cash from the ATM. In such situation the ATM may play a functional role. However, the purchase of goods is not usually considered the proper function of an ATM because it does not correspond to the lowest level of abstraction in the chain of events. Hence, the functional boundary of the ATM is usually established at the interface between the device and its user withdrawing the money. However, such delimitation may change given a different stakeholder perspective.

An analysis of this problem has been provided by Crilly [86], where the distinction conventionally drawn between devices and their environments has been replaced in favor of a more general 'systems' approach. According to this, any device can be considered as a system in itself, nested within a series of larger super-systems. Each system in the nested structure can be described as having both *endogenous* and *exogenous* functions. An

endogenous function refers to the functional role played by parts of the system, i.e. its sub-systems, with respect to the system itself, i.e. their immediate functional whole. This corresponds approximately to a device-centric perspective in FR. An exogenous function in turn refers to the functional role played by a system with respect to at least one of its super-systems. Thus, the logic of structural nesting of systems leads to a logic of function propagation, where any system may perform exogenous functions across multiple levels of abstraction.

Illustration 4.9 shows a diagrammatic representation of the nesting structure of systems and the propagation of exogenous functions at multiple levels. Here, the example of the ATM is further elaborated by Crilly to show the functional perspectives of two different super-systems, a bank and a shopping mall where the ATM is located. From the point of view of the bank, the exogenous function of the ATM is to satisfy the need of its clients to withdraw cash. From the point of view of the managers of a shopping mall though, the exogenous function of the ATM is to satisfy the need for exchanges with their many stores (i.e. support sales). Notice that the real value of the ATM is given by the role it plays at the highest levels of abstraction, pertaining to the services and business exchanges enabled by its local functions. Given other forms of payment, the value of the ATM and its internal technical functions is obviously reduced.

In his work, Crilly proposes an a more comprehensive characterization of functions by taking in consideration multiple levels of abstraction at the same time. For that purpose, not only the distinction between 'devices' and 'environments' is replaced in favor of a common characterization in terms of 'systems', but also the distinction between 'functions', 'needs' and 'goals' is replaced by treating all in functional terms. According to him, this allows to reconcile different interpretations of function found in the literature, providing a necessary clarification regarding the validity of global functions and their causal relationship with low-level technical functions [87]. This relationship is particularly relevant in the context of socio-technical systems, where the value of technical systems stems from their ability to contribute to the achievement of high-level global functions that are often non-technical in nature [85]. In this context, the representation of functional roles propagating across

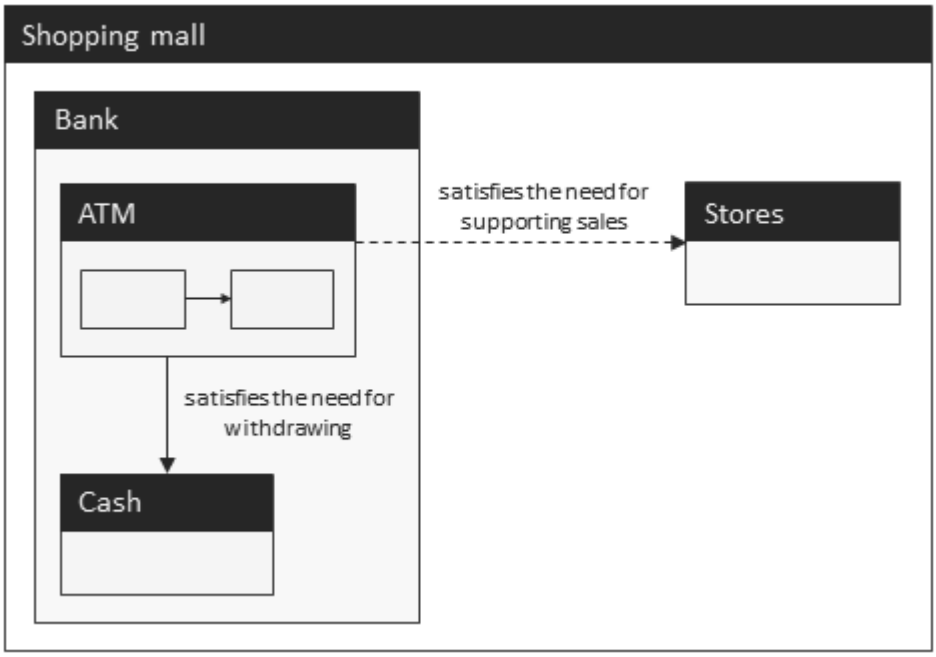

Figure 4.9: Endogenous technical functions of ATM and propagation of exogenous functions at multiple levels of abstraction. Adapted from [86].

multiple levels of abstraction is a critical challenge that remains to a large extent unresolved.

4.3.5 Roles and patterns of interactions

An additional problem in the description of behavioral interactions and the proliferation of functional roles is the evolving nature of socio-technical systems, which tends to change both during the design process, as well as during implementation and use. For instance, new functional roles can be ascribed on later stages of a system life-cycle, sometimes even opportunistically, according to emerging needs and modes of deployment.

For this same reason, the description of a mode of deployment is an evolving process as well, subject to constant reformulation due to continuous discovery of new goals and interactions [118]. Furthermore, multiple modes of deployment may exist for the same system or system component, depending on the stakeholders involved, and their overlapping sets of goals and needs. This implies that functional roles may exist that are the result of either intended effects or unintended side-effects. In such cases, functional conflicts can only be assessed upon identification of all functional perspectives involved, as they stem

from explicit expressions of stakeholders' goals and needs. For instance, a role considered positive by some stakeholders (e.g. amenities for office employees), may be deemed negative by others (e.g. liability or additional costs for managers).

Anticipation of conflicts among different functional roles is often not possible from the outset of a project, in part because modes of deployment are still unknown. Instead, timely identification and resolution of conflicts requires functional reasoning through multiple and continuously changing environment-centric viewpoints. These viewpoints provide the informational context from which inference about functional roles and functional conflicts can be made, according to the causal mechanisms and effects described in underlying functional models. In this regard, it is important to emphasize that roles are neutral with respect intent. Only statements of need or desire expressed by some intentional agent, i.e. a stakeholder, allows qualification of roles as either positive or negative.

While this may be trivial in simple design situations with few functional requirements, it becomes highly problematic when complexity increases due to proliferation of objectives over the life-cycle of a system. This is because of two reasons. First, from a qualitative perspective, and as discussed previously, a role deemed negative by someone may be perfectly desirable for someone else. Second, from a quantitative viewpoint, failure to acknowledge negative roles as indeed having a *functional participation* may lead to model incompleteness. To clarify the point it is useful to refer to another meaning of function presented by Chandrasekaran and Josephson, namely, the meaning of *function as behavioral constraint* (discussed in subsection 4.2.2.4).

This particular meaning relates function to *quantification of performance*. This involves the specification of constraints over measurable properties of effects playing functional roles. Hence, *function as behavioral constraint* can be interpreted as a specialization of the more general form *function as role*, in which more information is added to the specification. This is similar to the characterization of performance requirements as specialization of functional requirements, proposed by Augenbroe [16].

This specialization allows, among other things, the classification of artifacts with similar functionally according to some performance criteria. For example, while all street-legal cars

are designed to transport people, only those meeting certain measure of fuel consumption can be classified as fuel efficient. Given a different criteria, say acceleration, other classifications may follow. This in turn requires the specification of the conditions under which performance is quantified.

In design tasks involving quantitative prediction of performance (e.g. simulation), modeling all relevant roles, positive or not, is crucial to establish the structure of interactions that may have an impact on performance. Increasingly, this task involves assessment of uncertainty through sensitivity analysis and other methods of uncertainty quantification that rely on levels of model completeness. In this case, failure to include all relevant roles and the structure of their interactions should reduce the level of confidence on the predictions of a test or simulation model [320].

A similar interpretation of the meaning of function as a role is provided by Kitamura et al. [199]. According to these authors, a function is a *role played by a behavior in a teleological context*, where the teleological context corresponds to a mode of deployment under a particular environment-centric viewpoint. Notice that functional roles are played by output behaviors through their *effects* in the environment, and not by artifacts themselves. In Kitamura, the latter are said to play a *function-performer role* ([199], pg. 241) .

This account supports the idea that a side-effect normally considered as playing negative role may in fact play a positive role under special circumstances, either because the side-effect is beneficial in itself, or because it contributes to the improvement of performance associated to an explicitly intended functional effect.

This situation can be found in the context of buildings, considered as examples of socio-technical systems. For instance, heat gains due to sunlight exposure during summer are usually considered negative side-effects of components such as windows and other openings. However, it is not uncommon to find cases where this side-effect may contribute to regulate over-cooled spaces, especially within office buildings. While not ideal from an energy consumption standpoint, this may actually trigger some unique social behaviors. In such situations, the negative side-effect does not play just a role, but actually a positive functional role, albeit a rather ad-hoc one. Conversely, the intended effect of cooling down a

building, caused by the HVAC system, plays a negative functional role. But again, it all depends on whom you ask, and under what circumstances.

Therefore, the dichotomy or ambivalence between negative or positive roles can only be clarified by the specification of different modes of deployment, and according to the explicit assertion of priorities regarding needs and goals of stakeholders involved. However, the consideration of all relevant effects leads to the intractable problem of having to model all of them, along with every single possible interaction with the environment in which they play a role. Nonetheless, this does not need to be case. In practice, there are well known *patterns of interactions* that tend to recur for certain building types and technologies under certain general conditions. The over-cooling effect mentioned above is an example that sometimes result from unbalanced air supply to the different zones connected to an air handling unit. As side-effect, the over-cooling of spaces may contribute to a reduction of relative humidity, which in turn may conflict with the functional role of keeping air temperature within a comfortable range [211]. Such negative interaction can be considered a pattern that is known to apply given certain combinations of building types, systems integration strategies and environmental conditions [264].

Thus, the identification of all effects and roles that are relevant for the assessment of building performance *usually* conform to some known general pattern of interaction. Meaningful patterns can be inferred from particular combinations of building systems and elements of the environment, in relation to the main functional needs to be met. In Building Design, many of these combinations are rather stereotypical and are relatively well documented. The challenge however, is how to formalize such patterns in an abstract, generic way so that they can be reused in different design scenarios where multi-criteria evaluation of performance is key.

While attempts have been made to prescribe combinations of building elements that have participation in the satisfaction of functional requirements, the resulting models proved to be too brittle to handle variations and exceptions that normally occur in Building Design. The main problem is the reliance on fixed, single hierarchies of nested systems and components without accounting for cross-cutting participation of elements through propagation of their

side-effects.

Therefore, the main requisite for the formulation of a pattern of interaction is its ability to account for such cross-cutting participation, so that elements playing a role in the satisfaction of a function can be identified from any building sub-system, and at any level of abstraction. A collection of such elements corresponds to a cross-cutting aggregation called the *aspect system* of such function. In this way, an *aspect model* can be considered a functional model that captures both the aggregation and the functional criteria behind the aggregation.

4.4 Towards a formal representation of aspect systems

It is not completely unusual to hear stories about HVAC systems over-cooling a building because of heat vents of copy machines carelessly placed next to thermostat sensors. Anecdotes like these serve to illustrate the concept of aspect systems as aggregations of two or more apparently disconnected subsystems which are nevertheless functionally related, albeit unintentionally, by means of negative interactions. While many of these interactions are result of the idiosyncratic way people often use different artifacts and systems, other interactions are deeply ingrained in the structure of the design itself, as result of decisions made by multiple actors during the design process.

For instance, it is tempting to think that in the example of the HVAC system discussed in the previous subsection, the conflict between the functional role of 'reducing-humidity' and the functional role of 'keeping-the-temperature-within-a-range' is internal to the HVAC system, i.e. a device-centric viewpoint. Certainly, excessive humidity is an input coming from the larger operational environment. However, it is important to consider the source of the humidity in the first place. It could be, for instance, because of water vapor coming from a kitchen that is open for office workers, in addition to the transpiration produced by indoor plants. Hence, the internal conflict of the HVAC system is just a local manifestation of conflicts taking place elsewhere among different subsystems. Implicitly though, the real conflict is between competing environment-centric functions, and ultimately, between different stakeholders' goals and needs. Open kitchens for example are sometimes added

to office spaces to promote socialization among employees, while indoor plants are used for abatement of air pollution [63], or because their psychological effects may contribute to the improvement of productivity [259].

For this reason, functional verification and performance evaluation always need to take place in the context of other, possibly competing, functional criteria. In the example, this means that both the kitchen and indoor plants need to be part of the *aspect system* associated with the evaluation of thermal comfort, for their interactions are functionally relevant [16]. Assuming that the kitchen and the plants satisfy their own specific set of functional goals, a multi-criteria evaluation and decision-making process will need to be put in place for the resolution of potential conflicts.

In this regard, the concept of aspect systems provides an useful abstraction. First, it captures all elements having a relevant participation in the satisfaction of a function, providing the basis for a more informed process of performance evaluation. Second, it allows the identification of cross-cutting interactions, whenever a same element belongs to more than one aspect system. In this way, different strategies for conflict resolution can be tested and design trade-offs negotiated among stakeholders involved.

Because of this, functional modeling should support the incremental elucidation of aspect systems, capturing the evolving structure of behavioral interactions during different stages of the design process. This involves by definition the ability to drawn inferences under conditions of partial or incomplete knowledge, especially during early stages of design. For that purpose, providing reasoning capabilities under an Open World Assumption (OWA) becomes an important consideration to handle uncertainty. Figure 4.10 illustrates the envisioned process for elucidation of aspect systems under OWA, and the use of interaction patterns to support the inferences required.

There are two main scenarios in which uncertainty needs to be handled, according to the criteria discussed by Falkenhainer and presented in Figure 4.8. These include uncertainty regarding unknown roles of typed instances, and uncertainty regarding instance types (e.g. untyped entities). The former is depicted in the figure by the role $\mathtt{R_m}$. which can be inferred according to an interaction pattern associated to a known side-effect of typed element $\mathtt{E_m}$.

Unknown elements corresponding to untyped entities, their effects and possible roles may be inferred under OWA from structural relations implicit in the broader context or asserted by designers (e.g. entity E_u and role R_u).

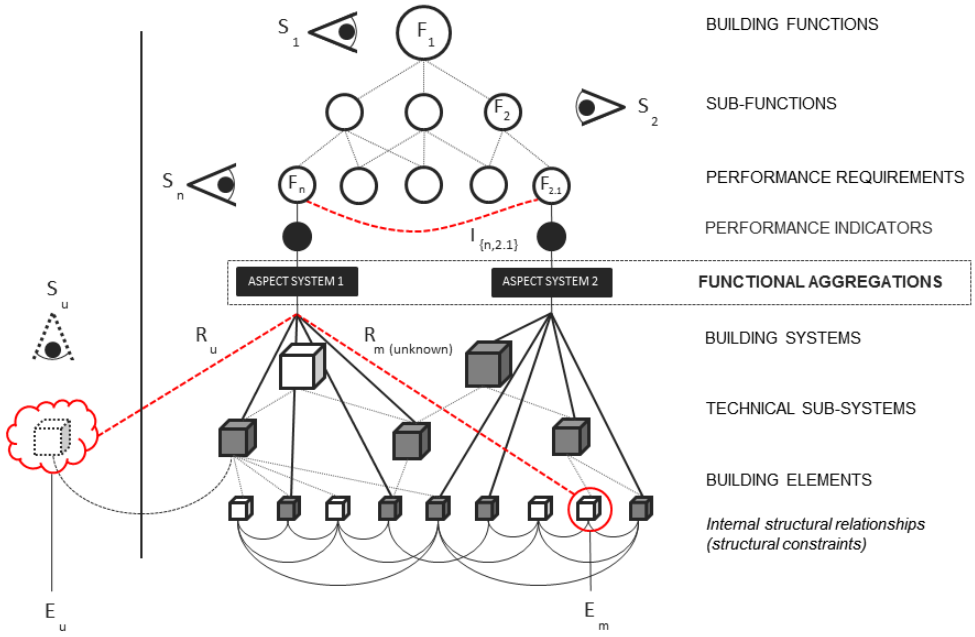

Figure 4.10: Unknown roles played by side-effects of known entities may be inferred by general patterns of interaction. Elucidation of aspect systems allows identification of potential synergies or conflicts across different functions. OWA allows plausible reasoning over untyped entities.

A functional interaction between different systems is denoted in the figure by the arc $I_{\{n,\ 2.1\}}$. This interaction may be positive or synergistic. For example, plant transpiration may contribute to thermal comfort by increasing relative humidity in a dry environment. In this way plants can contribute to the function of a HVAC system. In other circumstances, the interaction may be negative, which leads to a potential conflict between functions. More specifically, between different environment-centric functional viewpoints. In the case of interaction $I_{\{n,\ 2.1\}}$, the functional viewpoints involved are depicted by S_1 and S_n. Typically, a functional viewpoint can be associated to a group of relevant stakeholders.

Notice that the role R_u of element E_u in function F_n can only be considered a potential conflict insofar a functional viewpoint S_u is made explicit. Otherwise it should be considered a behavioral constraint enforced by the environment.

A functional modeling framework for Building Design should support inference of different functional roles and their implications across multiple levels of abstraction. Eventually, it should also provide capabilities to facilitate multi-criteria evaluation and decision-making, improving in this way the process of interdisciplinary collaboration in design. For that purpose, the notion of aspect systems provides an useful abstraction to capture the cross-cutting structure of functional aggregations that cannot be described by any single-domain compositional hierarchy (e.g. building technical systems discussed in 3.3.2).

In this regard, the notion of an 'aspect' discussed here shares similarities with the principle of aspect-oriented programming (AOP), in the sense that an aspect denotes a cross-cutting functionality involving different parts of a program that cannot be represented by conventional compositional methods. In AOP, an aspect is an abstraction that encapsulates both the specification of a cross-cutting function (called 'concern' in AOP) as well as mechanisms for 'weaving' functional inter-dependencies scattered across different types and levels of abstraction [241, 242].

The encapsulation of function specifications along with methods for 'weaving' of functional inter-dependencies suggests a viable approach for a model-based representation of aspect systems capable of supporting inference of functional interactions impacting performance. Such a model-based representation of an aspect system is compatible with the concept aspect model discussed by Augenbroe in the context of building performance simulation [16].

Ideally, aspect models should be abstracted into modular units of reusable functional knowledge, to be applied in different design situations, and in service of different tasks, according to one of the principles for functional modeling proposed by Goel (see principle 7 in [154]). The expectation of reusability relies on three observations that apply, in general at least, to most buildings:

- Buildings are usually required to provide a common set of general high-level functions (e.g. thermal, visual and acoustic comfort, privacy, accessibility, safety, ergonomics, etc.). Variations occur mostly at the performance level, regarding specification of measurable outcomes, and according to different priorities.

- Second, while the number of possible design configurations involving different building subsystems is relatively high, strategies for the integration of these subsystems tend to fall under categories that are well known in practice.

- Third, for each type of systems integration strategy, and according to general building functions, certain types of behavioral interactions tend to recur. These behavioral interactions can be captured abstractly as reusable patterns, i.e. abstract interaction patterns.

In summary, aspect systems can be defined as cross-cutting aggregations of different building parts, possibly from multiple building subsystems, and from any level of abstraction, that are known to *usually* play a role in the satisfaction of an environment-centric function.

One of the challenges for the implementation of aspect systems is the formalization of the various meanings of function in a compatible manner. Since buildings integrate many different design disciplines, from architecture, engineering and construction, to interior and landscape design among others, the diversity of possible meanings may be considerable. An initial approach would be the formulation of compatible subsets allowing more general forms of functional 'weaving'. This is especially relevant in the case of device-centric descriptions of functionality, which tend to be highly domain-specific. Likewise, inference of high-level functional interactions across different systems can only happen if environment-centric descriptions of functions are also specified in compatible terms.

The analysis of the Functional Representation (FR) scheme suggests a general framework for the unification of different meanings of function usually found in engineering design. In many respects, the dual characterization of function as intended *role* (of an effect), as well as *behavioral constraint* (over some property of the effect) is general enough as to cover some of the meanings found in other design disciplines, such as architecture, industrial design, interior design, etc. including aspects related to human and social behavior, as they relate to the built environment.

More recent efforts have focused on providing an unifying definition covering both technical as well as biological functions grounded on the notion of function as role in relation to different types of functional contexts [235]. Efforts also have been made to reconcile the meaning of artifact function across different levels of abstraction, in an attempt to integrate the description of technical functions with high-level functions that are not necessarily technical, including ergonomic, social, and aesthetic functions [85]. In all these cases, the notions of role, and function as intended role, offer a common link to support, in principle at least, compatibility between different functional interpretations and views. In this regard, additional abstractions will be required to support mappings across views, including an operational characterization of aspect systems representing functional contexts of interest.

In attempt to provide a model-based characterization of aspect systems, the research adopts the formalization of the Functional Representation scheme, developed by Borgo et al. under the DOLCE framework [27]. This formalization supports the notions of roles and functional roles through the introduction of a series of new ontological commitments, namely the category of *perdurants* describing processes and effects, and the participation relation between artifacts and perdurants.

This formalization will be presented in the next chapter, as part of the theoretical background supporting the research hypothesis and methodology.

CHAPTER V

HYPOTHESIS AND METHODOLOGY

The general problem of providing computational support for interdisciplinary collaboration in Building Design, is addressed in this research by proposing a theoretical framework for the development of functional modeling capabilities not currently available in BIM applications.

Among the various design tasks that could benefit from these capabilities, this research focuses specifically in the problem of identification of behavioral side-effects and cross-cutting interactions that emerge among various building sub-systems, mostly during the design process.

The relevance of this problem stems from the negative implications that many unanticipated side-effects and interactions have in the satisfaction of different functional requirements, which may span the entire life-cycle of a building. In particular, unanticipated side-effects and interactions constitute an important source of uncertainty, affecting the level of confidence regarding assessments of performance, and the delivery of value to stakeholders.

At the crux of this problem is the lack of computational methods to formally represent the various functional and behavioral dimensions of buildings. More specifically, there is no operational formalism to support a machine-readable specification of functionality with associated performance criteria. This affects both the 'demand' side, concerned with the description of functional goals and requirements, as well as the 'supply' side, concerned with the functional characterization of building elements capable of satisfying such goals and requirements. Because of this limitation, traceability of satisfaction relations during the design process has to be done and maintained manually. Obviously, this is time-consuming and prone to error, leading inevitably to several consistency problems that are pervasive across the building industry. At the core, the problem lies in the lack of traceability for cross-cutting interactions impacting multiple levels of buildings' functionality.

For this reason a robust functional modeling framework for Building Design is needed,

so that integration of various functional perspectives, involving different design disciplines and stakeholders could be better supported. For that purpose, a new layer of semantics is proposed to complement current information modeling schemes underlying BIM applications used in the AEC industry. The limitations of these, particularly in their ability to support multi-criteria design evaluation and decision-making, have been analyzed and presented in previous chapters.

The framework proposed here is grounded on theoretical models of functional representation and reasoning developed in the areas of Artificial Intelligence and Engineering Design. It also draws from research in the area of Applied Ontology, related to theories and methods for the formalization of functional meaning. These lines of research offer a theoretical and methodological foundation to address the representation of functionality in the context of socio-technical systems such as buildings, where multiple functions need to be satisfied by means of careful integration of different types of systems.

At the implementation level, a functional modeling framework as envisioned in this research would become an additional level of semantics on top of current design representation schemes, particularly those associated with BIM applications, such as IFC. Given the complexity of dependencies associated to proprietary or legacy building data models currently use, such a semantic layer should be developed independently from current BIM models and applications, but in a compatible fashion, to be integrated incrementally on an as-need basis. In that regard, the independence of layer would not only avoid unnecessary complexities of retrofitting exiting building data models, but it would also benefit from other concurrent efforts towards semantic unification of various function types.

The review of the literature available in the areas of CAD, BIM, Artificial Intelligence and Applied Ontology provides a necessary historical context to situate the potential contribution of this research, as well as a number of principles and guidelines for the development of a framework addressing both theoretical and practical aspects. Some of these frameworks, such as the FR and SBF have a common theoretical basis, sharing similar approaches regarding the characterization of function and behavior at different levels of abstraction.

The analysis of these two schemes in the previous chapter motivated the adoption of

several important ideas. Among these, the notions of side-effect and side-function, as well as the concept of generic design patterns introduced in the SBF have been particularly relevant. In the case of FR, the characterization of **function as a role** provides a strategy to address a range of interpretations found across disciplinary building domains. Moreover, the concept of mode of deployment suggests a viable approach for the formal characterization of multiple functional perspectives, at various levels of abstraction.

Finally, with formalization of the Functional Representation schema under the DOLCE ontological framework, a more general model has been provided, with direct implications in the formulation of the research hypothesis.

Before introducing the final hypothesis however, a review of additional theoretical background is required. For this reason, a preliminary hypothesis is presented first. This will be followed by a discussion on the ontological limitations of current CAD / BIM applications, and an overview of the DOLCE foundation ontology. Then, the formalization of FR under DOLCE is discussed, based on a small subset of formal definitions considered relevant for the goal of this research. Finally, the hypothesis is presented, followed by the research methodology.

5.1 Preliminary hypothesis and background

Within the context of the general problem, and the potentially wide range of design tasks that a functional modeling framework could cover, the research focuses specifically in the problem of elucidation of *aspect systems*, introduced in Section 1.4, Chapter 1. Aspect Systems have been informally defined as cross-cutting aggregations of different building parts, at different levels of the composition hierarchy that participate in the satisfaction of a building function. The identification of these parts is predicated on the identification of the effects they cause in the environment, either intended or not by design. These effects of can be described a priori as the behavioral space of parts involved, or the supply side of functions, as discussed previously in subsection 3.4.5. Given that the demand side of functions can also be associated to a behavioral space, aspect systems can be inferred by the overlapping of both spaces.

Under this particular perspective, the specific requirements for a functional modeling framework as envisioned here are:

(i) To support explicit description of known behavioral effects and interactions, intended or not, that tend to recur across different building subsystems at different levels of abstraction.

(ii) To support specification of behavioral constraints over effects and interactions from multiple functional viewpoints (performance aspects), spanning the entire life-cycle of the building.

(iii) To provide inferences over implicit cross-cutting interactions that could lead to conflicts (or synergies) among different functional viewpoints, spanning the entire life-cycle of the building.

The preliminary hypothesis of this research is grounded on the multiple characterizations of function proposed by Chandrasekaran and Josephson, which provides a necessary vocabulary to address the requirements above. This is because different functional viewpoints that need to be covered in building design cannot be captured by a single universal definition. Moreover, the need to support cross-cutting functional viewpoints, especially at different levels of abstraction, will require the various meanings of function to have compatible interpretations.

In FR, the characterizations of function as (i) *behavioral constraint*, (ii) *intended effect on the environment*, and (iii) *function as intended role*; as well as the notion of *mode of deployment* offer a conceptual framework on which this requirement of semantic compatibility could be satisfied. This allows to formulation of the research hypothesis, based on the following two criteria:

1. **Functional integration:** Co-participation of two or more building elements, possibly from different building sub-systems, and possibly from different levels of structural composition, in the satisfaction of a same function, and independently of the intention

of a stakeholder regarding the co-participation, is a necessary condition for these elements to be included as part of the aspect system of the function.

2. **Multi-functionality:** Participation of the same building element in multiple functions, independently of the intention of a stakeholder regarding the multiple participation, is a necessary condition for such element to be included as part of the aspect system of each of those multiple functions.

These two criteria capture two of the main features of a systems integration problem, including those associated to design and operation of buildings. Thus, the 'functional integration' criterion indicates the fact that high-level functions can only be satisfied based on careful integration of parts from different subsystems, and at multiple levels of composition. The 'multi-functionality' criterion refers to the fact that elements can be multifunctional if they have behavioral side-effects that contribute, in a synergistic manner, to the satisfaction of other functions. Conversely, behavioral side-effects can be detrimental, which could potentially lead to functional conflicts across different building subsystems. As will be discussed later in the formalization of FR in DOLCE, multi-functionality of a component does not mean that such component satisfies, by itself, multiple functions, nor that it contributes in a positive manner to their satisfaction. Rather, it means that it participates in the interactions that cause the satisfaction of multiple functions. This in important from an environment-centric perspective due to following reasons:

- It is the *mode of deployment* in which various components participate that satisfies a function, not technical systems or components themselves.
- The assessment of 'value' of a given functional participation, i.e. either positive or negative, depends on the mode of deployment, and intentionality of stakeholders involved.

5.1.1 Relevance for systems integration and design-analysis integration

Successful systems integration require careful traceability of cross-cutting behavioral interactions and functional inter-dependencies. However, this can be extremely difficult without

computational support, especially in complex, interdisciplinary projects under continuously reformulation of domain-specific requirements.

Figure 5.1 illustrates the problem, in relation to the criteria of functional integration and multi-functionality in the context of the photovoltaic racking system design discussed in Chapter 1, section 1.2.2.The horizontal axis in the Figure indicates the aspect systems of the functional requirements described on the left column. These are related to ergonomic aspects of construction and installation of photovoltaic systems. The vertical axis indicates the parthood in various aspect systems for parts on the top row. Parthood in an aspect system implies participation in an effect playing a functional role.

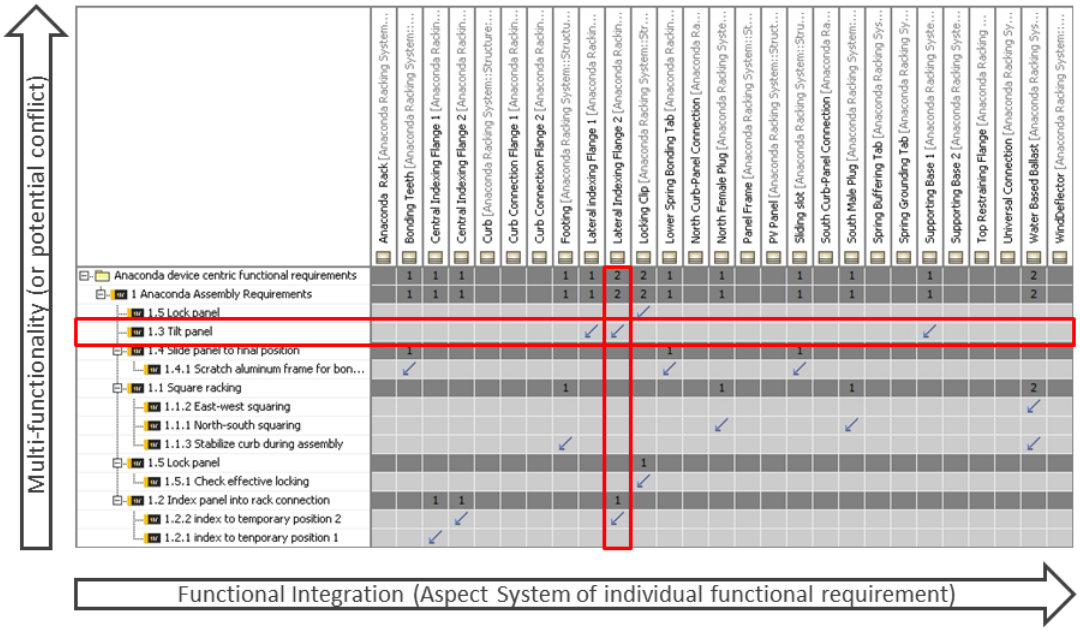

Figure 5.1: Functional integration and multi-functionality depends on the elucidation of aspect systems.

This dependency matrix was developed in SysML, based on a customization of the SysML requirements diagram notation with a Domain Specific Language (DSL) called SBReq. This customization allowed explicit assertion of participation relations by means of a behavioral model associated to each individual functional requirement [70]. Figure 5.2 illustrates the specification of various functional requirements associated to structural

performance of solar racking systems. Countering wind forces is a sub-function that is specified in terms of the negative effects potentially caused by wind, including drag, vibration and uplift. New participation relations had to be introduced manually in the behavioral model (top of Figure 5.2), and updated automatically in the functional matrix. However, the model did not provide automatically derive indirect participation and parthood (e.g. by means of inheritance or transitivity), since this requires reasoning capabilities not directly supported in SysML or available SysML editors. This limitation caused difficulties whenever changes were made to the design, often as result of changes or refinements in the specification of functional requirements.

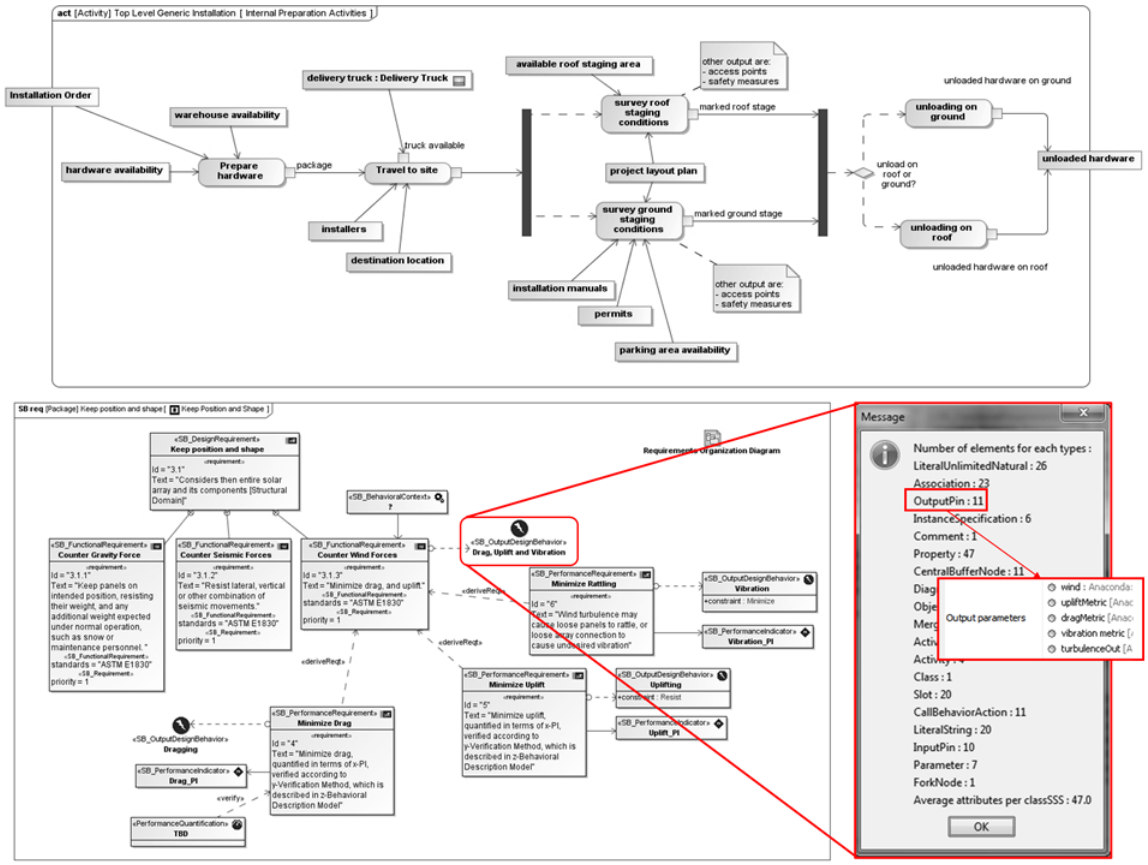

Figure 5.2: Behavioral model in SysML associated to a functional requirement. aspect system results by participation relations asserted in the behavioral model (top side). Output metrics on interest (right side).

For this reason, functional modeling and reasoning should support the identification of interactions across different functional viewpoints that are fundamentally incremental and

variable, as well as trade-off analysis and decision-making activities that are fundamentally collaborative, i.e. temporally and geographically distributed. This could be automated, partially at least, if behavioral models are linked to design models according to a formal, machine-readable language. Because the design process always occur in conditions of partial knowledge, embedded reasoning capabilities should be able to provide support under OWA.

It might be argued, however, that performance simulation models (e.g. energy simulation) already provide such a layer, with sophisticated, state-of-the-art behavioral models, making this attempt somewhat redundant and probably unnecessary. The problem however is that behavioral models underlying performance simulation and other analytical models have a much narrower focus within the larger context of a design process. In general, they serve the task of performance verification within domain-specific boundaries, capturing only a snapshot of the entire design process, and without much coordination with other performance aspects and functional perspectives.

Under the more recent Design of Experiments (DoE) methodology, cross-cutting behavioral interactions and inter-dependencies are explicitly asserted in behavioral models by means of parametric relationships [263, 275]. However, the underlying causal mechanisms have to be known beforehand, and assumed to remain stable over the course of the design process. This makes this approach fundamentally brittle in the face of design changes and evolving specification of functional requirements.

Since behavioral models underlying simulation or other experimental procedures are often not formally linked to requirements specification, the implications of changes in the specification are not transparent to other stakeholders involved. More recent attempts have been made to integrate specification of requirements with structural and behavioral models under a common modeling framework based on the Systems Modeling Language (SysML) [145, 144]. This integration under a centralized model would facilitate, in theory at least, the traceability of various relationships, allowing better control and management over design changes across multiple domain-specific views. Nevertheless, given the fact that SysML is not a formal language, semantic ambiguity may arise, along with different types of model inconsistencies [175, 174].

Moreover, traceability, parametric propagation and derivation of new information in SysML is limited, by operating, albeit implicitly, under the conditions of a Close World Assumption (CWA). Because of this, all entities playing functional roles are assumed to be known a priori. The lack of formal characterization of entities in SysML hinders the possibility of information processing under conditions of partial knowledge. This is particularly problematic when traceability of cross-cutting system interactions and inter-dependencies is needed.

It is this specific type of problem that the functional modeling framework proposed tries to address. In doing so, it is envisioned that the representational gap or asymmetry currently in place between design and analytical models could be reduced or eliminated. Eventually, this could lead to a more systematic, process-driven integration of design and analysis task in the context of multi-criteria, performance-based design.

To support a more seamless integration, it is necessary to provide the design 'side' with explicit and more robust characterization of behavioral properties, which was identified as one of the most important limitations in current building models (3). Notice that the role of this behavioral characterization is neither to enable a simplified, user-friendly simulation nor a shortcut to verification methods, but rather to enable (i) the formalization of meaningful input models for analytical procedures; and (ii) to make these input models the very subject matter of interdisciplinary design integration.

These input models are models of the aspect system that is relevant from a functional perspective, and for which a quantification of performance is in required. In other words, an aspect model is a model-based representation of an aspect system, especially tailored as input model for specific analysis applications, as discussed in 1.4. The introduction of an additional layer of representation depicting the functional and behavioral dimensions of a system are intended to facilitate the elucidation of which parts are participating in the fulfillment of a functional requirement and their impact in different levels of performance.

Figure 5.3 illustrates the role envisioned for a functional modeling framework for design-analysis integration under a multi-criteria Building Design process. First, it should enable a formal, model-based description of functions and their possible specializations in the form of

performance requirements. Given a required function, and the behavioral characterization of design components in terms of their environmental effects (i.e. output behaviors), aspect systems could be initially asserted by available knowledge. Aspect models are model views of aspect systems used as input for analysis tasks. For each required function, one or more Aspect Models may be defined.

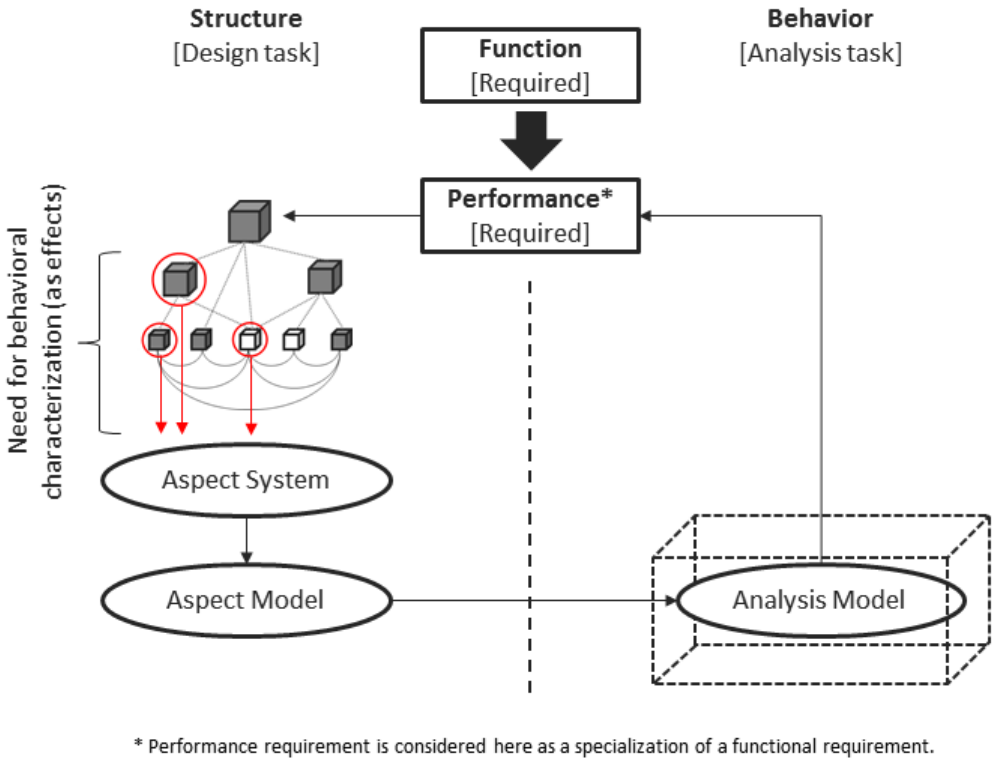

Figure 5.3: Functional modeling for design analysis integration underlying interdisciplinary collaboration. Model-based description of functional / performance requirements provides operational interface between design and analysis models.

This capability is not readily available by current building modeling schemes, including IFC, and can only be deployed by intensive customization and point-solution implementations. Additionally, and besides the inference of cross-cutting behavioral interactions required for the elucidation of aspect systems, the proposed framework could support additional design tasks, including selection and retrieval of predefined components from libraries, that match the functional constraints of an aspect system. In a similar way, automatic

checking procedures could be implemented to verify the validity of instantiated BIM objects within the functional context of multiple aspect systems.

5.1.2 Ontological limitations of CAD / BIM applications

In order to provide this richer level of design semantics, it is necessary to recall the main limitations of current representation schemes, and the reason behind such limitations discussed in 1.1. From an ontological perspective, it is clear that most representation schemes used in CAD and BIM design applications only commit explicitly with the existence of structural entities. Behavioral aspects are referred indirectly through the use of structural abstractions used as representational surrogates 3.4. These might range from topological and geometric properties of elements (e.g. dimensions of a corridor as proxy for circulation behavior) to material properties as abstraction of how an element is *usually known to participate* in a behavior (e.g. thermal resistance value of a wall as proxy for how a material resists heat flows).

The critical problem is that the behavioral phenomena referred to by representational surrogates have an independent ontological status from the structural entities that participate in them. As such, it is reasonable to treat such phenomena as first-class entities, with their own set of properties and relationships. This requires new ontological commitments that are at the core of some of the foundational ontologies being applied to a wide range of domains, where unambiguous characterization of behaviors, activities, processes and events is paramount for the purpose of data modeling and interoperability. These include fields as diverse as biomedical research [289], security and defense systems[302], geospatial information systems [66, 309] and manufacturing [28], among others.

Figure 5.4 depicts the new ontological commitments associated to the general category of behavioral phenomena called *Perdurant*, also known as *Occurrent*. According to Smith, traditional information systems commit only to what he calls the *Aristotelian Quartet*, depicted by the red box on the left. It is clear that such is the case for CAD and BIM applications, where only structural entities and their dependent qualities (i.e. properties)

are being represented under the general categories of *Endurant* and *Quality*[1] respectively.

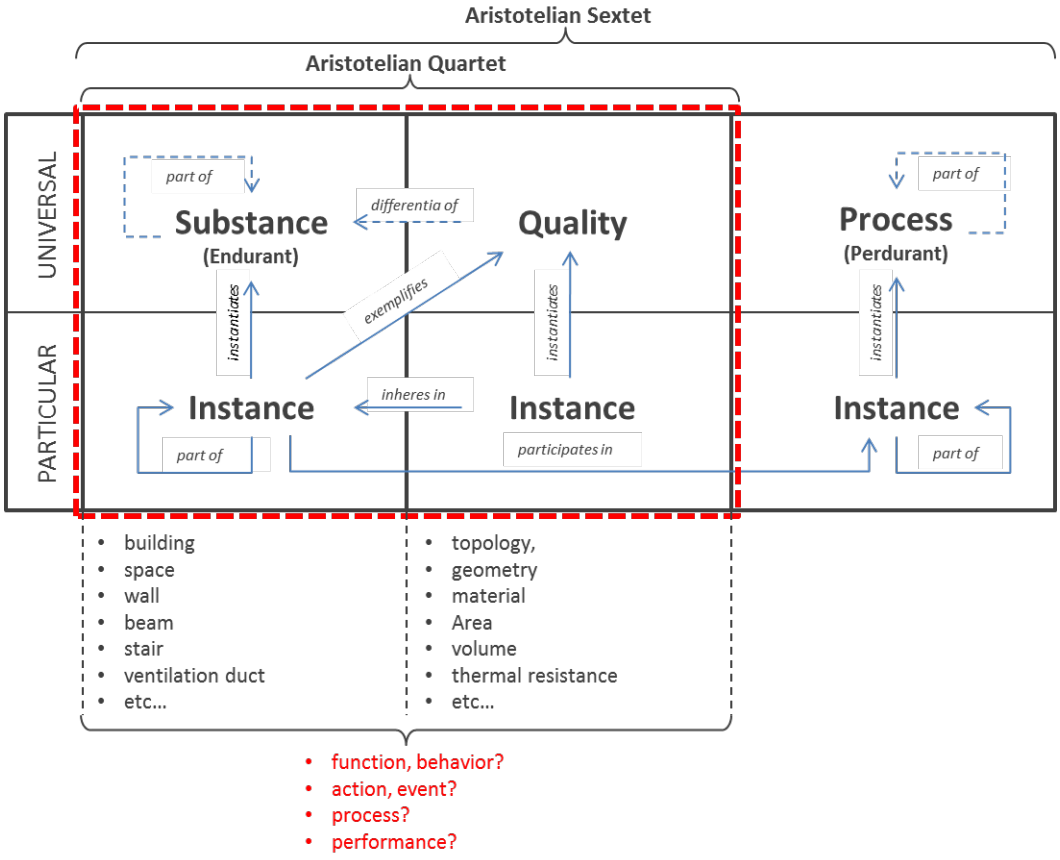

Figure 5.4: Aristotelian Quartet and expanded Aristotelian Sextet. Adapted from [288].

As it can be observed, semantically complex notions of function, behavior, processes, activities, events or performance are forced to fit into properties that are dependent on 'things', either natural, artificial or virtual. On the other hand, the new ontological commitment towards the category *Perdurant* provides additional expressivity and precision to describe phenomena that are fundamentally different from the 'qualities' of things referred by the category *Quality*. Particularly relevant now is the possibility of ascribing qualities that are specific to behavioral phenomena. Most notably, this new ontological commitment allows relationships such as merelogical parthood between perdurants [282], which has previously received only an ad-hoc treatment, particularly in IFC (See 3.3).

These new commitments lead to what Smith refers to as the *Aristotelian Sextet* [286,

[1]Quality is a general category for properties, but it has slightly different meanings in BFO and DOLCE.

288], at the core of foundation ontologies such as Basic Formal Ontology (BFO) and the Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE). The next section provides an overview of the basic common principles behind these foundation ontologies. Then, a more specific overview of DOLCE is provided, followed by an overview of selected axioms from DOLCE-FR. This will provide the necessary theoretical background for the formulation of the final research hypothesis and methodology.

5.1.3 Foundation ontologies

A foundation ontology is a representational artifact that provides a structure and a set of principles to organize knowledge. The purpose of such organizational structure is to support the description of the primitive and derived entities that are recognized as relevant within a domain. Among these entities, an ontology includes the formal treatment of relations as first-class entities. These include relations such as identity, difference, parthood, overlap, inheritance, dependence, participation and location [290]. An ontology is not a knowledge-base by itself, but it provides the formal semantic backbone for the development of knowledge-bases that operate with instance data that exemplify the semantics of ontological definitions.

One of the most commonly found definitions of ontology in the context of information systems in general, and Building Information Models in particular, comes from Gruber [165]. Such definition refers to an ontology as the "explicit specification of a conceptualization". However, some authors in the field of Applied Ontology disagree with this definition, despite its widespread adoption. For instance, Smith criticizes the notion of 'conceptualization' as stemming from an intellectual attitude of cultural relativism, according to which no objective description of reality is possible, but only multiple 'conceptual' interpretations. According to his criticism, interoperability problems arise due to the highly idiosyncratic nature of these interpretations [285, 289]. Guarino on the other hand, indicates that Gruber's definition is limited by the fact that it relies on an *extensional* account of ontological relations that hold within a domain, whereas an *intensional* characterization is required to convey with formal precision the logical meaning of these relations, independently of a

particular state of affairs [166].

In this context, a foundation or upper ontology is a special type of ontology, which attempts to rigorously describe the intensional meaning of a formal vocabulary, according to a particular conceptualization of the world [66]. In practical terms, this conceptualization reflects the world-view of a disciplinary domain, and therefore intensionality should be grounded on specific ontological commitments that are relevant to that domain.

In many design disciplines these commitments include terms such entity, feature, event, process, spatial and temporal location, and ontological relationships such as subsumption, parthood, participation, dependence and constitution. For that purpose, a foundation ontology provides a logical theory to formally describe and rigorously categorize the constituents of reality in a systematic manner, according to the most accepted interpretations within a domain of knowledge [284].

Some of the most important foundation ontologies are the Basic Formal Ontology (BFO) [14] and the Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) [231]. These are related by sharing a common set of top-level principles of categorization, which are based on the distinction between endurants and perdurants, *participation* of the endurants in perdurants, and parthood relations specific to perdurants, etc. However they differ in methodology, orientation and scope that reflect in part preferences over fields of application. From a methodologically viewpoint, BFO is strictly realist, in the sense that it attempts to describe concrete entities as *observed* in the world at the most general level possible, without committing to abstractions or conceptualizations deemed as domain-specific. DOLCE on the other hand emphasizes cognitive and linguistic aspects of domain-specific discourses, as the primary source for the formalization of *intended meaning*. In other words, DOLCE attempts to provide an ontological framework in which the meaning of terms, as normally used within a field of knowledge, becomes clearer. These includes commonsensical notions, concrete and abstract entities, agency and intentionality, among other concepts [163].

The implication of this distinction can be observed from the areas in which these two ontologies have been applied. Thus, BFO has a strong presence in biomedical and geospatial domains, whereas DOLCE is more concerned with domains dealing with technical and

social artifacts. By consequence, it seems that DOLCE is more aligned towards design and engineering applications, where the need for formal description of possibilities - and not just actualities- is paramount.

5.1.4 Descriptive Ontology for Linguistic and Cognitive Engineering

The Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) is an a foundation ontology with strong orientation towards engineering applications. This stems from its preference on cognitive aspects underlying the semantics of a domain of discourse, as opposed to a strictly realist approach. Thus, DOLCE aims to capture and formalize the ontological categories implicit in natural language and commonsense knowledge. This approach carries two important consequences. First, that DOLCE categories are situated at a mesoscopic level of objects, that is, at a scale which humans normally perceive or interact with. Second, DOLCE makes no claim about robustness or correctness against the state-of-the-art in scientific knowledge. Thus, DOLCE's categories are just formal descriptions that make explicit the conceptualization already in use within a domain of knowledge, independently of their scientific validity [27].

DOLCE focuses on *particulars*, and the ontological categories of particulars which differentiate among physical and abstract objects, events and qualities. The taxonomic structure of DOLCE is presented in Figure 5.5. The top category is `PARTICULAR`. The category `ENDURANT` comprises objects, both physical and abstract, portions of an object, such as features and lumps of matter. An endurant (also known as *continuant*) thus is defined as an entity that exists in full at any time it exists at all, persisting, continuing or enduring through time while maintaining its identity. Existence in full means that all proper parts essential to the identity of the entity are present at any given time.

The category `PERDURANT` on the other hand, comprises entities such as processes, events, actions or states. These entities never exist in totality whenever they exist, but can only exist in part at any given time. Thus, a perdurant (also known as *occurrent*) is an entity that unfolds itself in time or it is the instantaneous boundary part of such an unfolding entity. For example, 'running' is a perdurant, which has other perdurants as its parts, including

the instants when someone starts or stops running, among many other possible temporal parts. Therefore, the main criterion of differentiation between these two categories is their behavior *time*[166, 286].

Some examples of the category `ENDURANT` are a brick, a lump of concrete, a protrusion on a metal plate, holes in a mounting bracket, a mechanic device. Examples of `PERDURANT` are *laying* bricks, *mixing* concrete, *stamping* of the protrusion and the *drilling* of the bracket holes, the *functioning* of the mechanic device, among others.

`QUALITY` is the category of entities that we can perceive or measure. For example, shapes, colors, sizes, sounds, smells, textures, weights, intensities, to name just a few. Informally, and for practical purposes, qualities can be considered somewhat similar to properties, in the sense that their instantiation is dependent on the pre-existence of a bearer entity. In DOLCE however, qualities are not, strictly speaking, the same as properties. Qualities are *particulars*, while properties are considered *universals*. Instead, individual qualities can be better thought as *instantiated properties* [27].

Notice that not only endurants can have qualities, but perdurants as well. Qualities are related to either kind as long as they exist. Also, it is important to distinguish between a quality (e.g. the color of a specific rose), and its value (e.g. an instantiated RGB value). The former denotes a quality space, while latter refers to the specific position of the individual quality within such quality space [231].

5.1.5 FR Formalization under DOLCE / DOLCE-FR

This section introduces the First-Order Logic formalization of the Functional Representation scheme by Chandrasekaran and Josephson [76] according to the DOLCE foundation ontology, by Borgo et al. [27], referred here as DOLCE-FR. In this section only a subset of the axiomatization will be presented, along with few necessary ancillary definitions. For a more complete information, the reader can refer to the original documentation cited above.

The initial selection introduced here focuses on the characterization of *output behaviors*, which are considered sufficient to cover the two criteria of the preliminary hypothesis regarding elucidation of aspect systems, namely, the co-participation of endurants in the

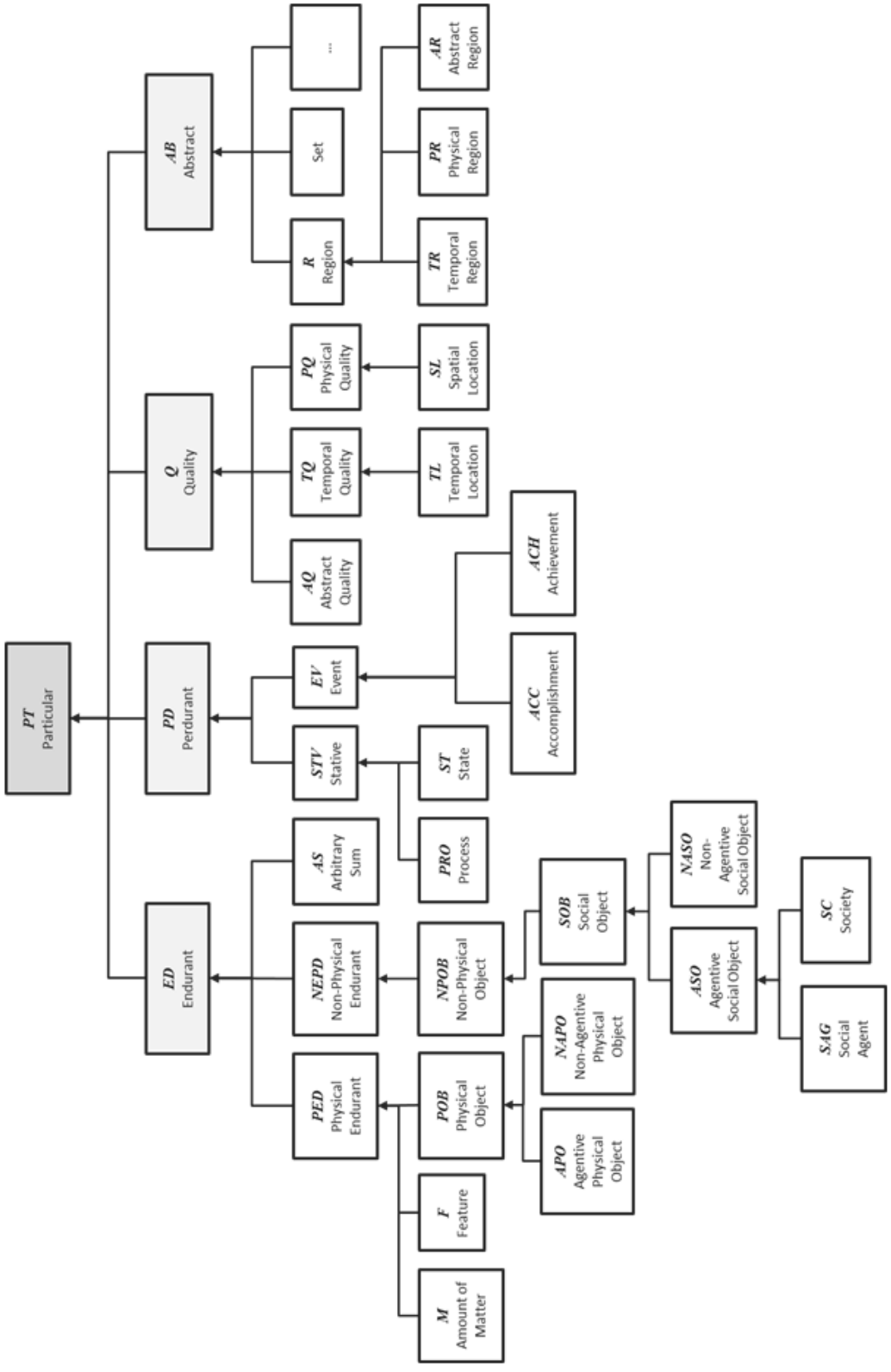

Figure 5.5: DOLCE taxonomic hierarchy. Adapted from [166].

same perdurant, or multiple participation of the same endurant in multiple perdurants that play functional role.

5.1.6 Mereological definitions

At the first stage, the research focuses on the translation of a subset of FR axioms from First-Order Logic into OWL-DL. The main objective of this translation is to support the first step in the process of elucidation of Aspect Systems, involving inference of behavioral interactions based on the principle of co-participation of different subsystems in the same perdurant. Such perdurant may be a process, event, state or activity that is explicitly intended by an agent (i.e. a stakeholder), through the specification of a functional / performance requirement.

However, and for the sake of simplicity, the criteria of intentionality of an agent is not strictly necessary, since teleology may exist without intentionality. Therefore, the only criteria that is initially required is:

(i) There are structural relationships between a technical artifact and some objects of the environment.

(ii) There are some processes, actions, states or events in which the artifacts and these objects from the environment may co-participate in.

These two conditions demand further clarification. First that structural relationships may stem from internal parts of artifacts and other objects of the environment. Therefore, parthood relationships are relevant to assess co-participation, given that behavioral interactions can be originated from side-effects produced at any level of the compositional hierarchy.

In order to formulate co-participation as criteria for elucidation of Aspect Systems, a few ancillary definitions related to mereological parthood are presented first, followed by a selection of FR axioms. In the original work, the intuitions behind the formalizations are explained with various examples. Here, examples are provided from situations that are often the subject matter of interdisciplinary Building Design.

The *proper part* relation is denoted formally by PP. It allows to describe the fact that a perdurant is a proper part of another. Here, the definition is presented without reference to temporality. For example, making a goal is a proper part of the soccer game, but no vice versa.

$$PP(x, y) := P(x, y) \wedge \neg P(y, x) \tag{1}$$

Mereological overlapping of two perdurants, formally O, occurs when another perdurant exists that is simultaneously part of both. For example, attack and defense actions in a soccer game overlap because the moving ball is part of both.

$$O(x, y) := \exists z(P(z, y) \wedge P(z, y) \tag{2}$$

In the context of building systems, overlapping as defined here indicates the possibility of dual participation of an object, artifact or system in the performance of more than one function. For instance, day lighting participates both in visual task performance and thermal performance due to heat gains.

Mereological *sum* $(+)$ and *fusion* (σ) are operators allowing special forms of composition that are relevant to perdurants. In the first case, a perdurant z is the sum of perdurants x and y if each part of x and y are also part of z, and if a perdurant w overlaps z, then it also overlaps either x or y.

$$x + y := \iota z \forall w(O(w, z) \leftrightarrow (O(w, x) \vee O(w, y))) \tag{3}$$

For example, the process of thermal balance in a room can be described in mereological terms as the sum of heating and cooling. Day lighting overlaps either one or the other.

The *fusion* operator σ extends the binary sum to all perdurants that exhibit property ϕ.

$$\sigma x \phi(x) := \iota z \forall y(O(y, z) \leftrightarrow \exists w(\phi(w) \wedge O(y, w))) \tag{4}$$

For example, the heating of an office space is the fusion of all heating cycles of a heater within such space (i.e. when the heater is activated). Clearly, the property can be defined as conjunction. Hence, the formula can be extended to include heating from all sources within the same space. Both definitions are functional, as indicated by the use of the *iota* in ιz. This means that exactly one z exists such that formulas (3) and (4) are satisfied. From these, a few more ancillary definitions follow, based on the work on mereology by Simons [282]. For brevity, and to further illustrate the notion of mereological fusion, the following theorem is presented:

$$\phi(x) \rightarrow P(x, \sigma y \phi(y)) \tag{5}$$

Intuitively, this allows an initial fusion to be expanded, by adding new perdurants exhibiting property ϕ. For instance, when new sources of heating are brought into the space. Since parthood is transitive, x itself might be a mereological fusion, say, of heating perdurants. Therefore, it is also part of the larger fusion of heat transfer processes taking place in the same office space, which need to be all considered for the accomplishment of thermal balance.

5.1.7 Behavior and participation relations

The relation *participation* in DOLCE describes the idea that an endurant a 'lives' during a period of time by participating in some perdurant p. This relation is formalized as follows:

$$PC(a, e, t) \tag{6}$$

Note that t may be just a part of the total duration of e. In this way parthood and participation are made clearly distinct, constraining the way perdurant and endurants can relate to each other [27]. Thus, parthood can only be established within the same categories, with no parthood admitted between endurants and perdurants.

The participation relation between endurants and perdurant indexed by time parameters allows for the description of situations involving temporary or complete participation.

A variation of this relation is the so-called "wholly participation", which characterizes the class of perdurants e in which an artifact a participates throughout the entirety of its life:

$$PC_{\mathrm{WH}}(a, e) \tag{7}$$

Thus, in the thermal balance example, different sources of heating and cooling wholly participate in the their specific heat transfer processes of heat gain and loss. Yet, they only partially participate during different time intervals in the overall fusion pertaining to thermal balance.

In DOLCE-FR, the behavior of an artifact is characterized as *"the specific way in which an artifact participates in some perdurant"*. Therefore, a behavior is treated as a *quality* of artifacts describing aspects of its participation, without having an independent, first-class status. This adds a level of 'articulation' allowing more flexibility in the way that behavioral phenomena can be modeled and processed. In the context of design, it helps to make more evident the inherent contribution that the artifact brings to the unfolding of the phenomena being described. This relation is formalized as primitive ternary relation:

$$Beh(a, e, b) \tag{8}$$

Here b can be interpreted as the behavior of artifact a in event e. Since the qualification of the participation depends on pairs of endurants and perdurants, binary and unary definitions can be derived, based on existential quantification over the excluded variables.

$$Beh(a, b) := \exists e Beh(a, e, b) \tag{9}$$

$$Beh(b) := \exists e, a Beh(a, e, b) \tag{10}$$

Since e might have other participating artifacts, these definitions suggest an initial framework for the description of behavioral interactions spanning across multiple viewpoints. This is especially relevant, given the fact that e might be represented as the sum

or fusion of other perdurants, from which cross-cutting interactions could be inferred. Such expectation is grounded on the condition of uniqueness imposed by a qualified participation, given by the following axioms:

$$Beh(a, e, b) \rightarrow PC(a, e, tm(e)) \tag{11}$$

$$Beh(a, e, b) \wedge Beh(a, e, b') \rightarrow b = b' \tag{12}$$

$$Beh(a, e, b) \wedge Beh(a, e, b') \rightarrow a = a'/wedgee = e' \tag{13}$$

$$PC(a, e, tm(e)) \rightarrow \exists b Beh(a, e, b) \tag{14}$$

The function $tm(e)$ in definitions (11) and (14) is the whole period of time during which perdurant e occurs, so that the participation of a in e is temporaly located. Therefore the participation can be qualified for that period, i.e. the behavior of *this* artifact instance a in e during a specific moment.

In order to describe not only actual behaviors, that is, what an artifact actually does during its life (or usually does), and the space of possible behaviors that are relevant from an engineering design perspective, DOLCE-FR introduces the class of *engineering perdurants*, formally EPD. This in turn is subsumed by the class of *generalized engineering perdurants*, formally $GEPD$. The latter allows to refer to behaviors that might be physical impossible, given current state of knowledge, but which are not illogical. The entire subsumption relation is stated as:

$$APD \subseteq EPD \subseteq GEPD \subseteq PD \tag{15}$$

With this definition, the domain of Beh can be constrained as follows:

$$Beh(a, e, b) \rightarrow TechArt(a) \wedge GEPD(e) \wedge B(b) \tag{16}$$

Where $TechArt(a)$ indicates that a is technical artifact. However, this does not need to be the case. For buildings and other socio-technical systems involving non-technical entities with some participation that is relevant to design, the more general categories of *Non-Agentive Physical Objects* ($NAPO$), or *Agentive-Physical Objects* (APO) is available (see Figure 5.5).

Other issues that requires consideration are the criteria of *coherence* and *minimality* regarding sum and fusion of perdurants, respectively. In DOLCE-FR, two or more perdurants are said to be coherent if their sum is a (physically) possibility. The criterion of minimality partitions the class of all generalized perdurants to allow the identification of various units of participation, that characterize the *life* of an endurant according to the criteria of actuality, possibility and generality introduced in (16). Thus, the *actual life* of endurant a, here assumed to be an artifact, is an perdurant of the class $APD(e)$. Formally, the notation of actual life of an artifact is $Alf(a, e)$, which is defined according to the following axiom:

$$Alf(a, e) := \exists e'(APD(e') \wedge PC_{\mathrm{WH}}(a, e')) \wedge e = \sigma y((APD(y) \wedge MIN(a, y)) \qquad (17)$$

Where $PC_{\mathrm{WH}}(a, e')$ indicates that a wholly participates during the entire span of the actual perdurant e', and that e is the fusion of all actual perdurants $APD(y)$ which are minimal to respect the class of generalized perdurants. From this, axioms for the *'possible life'* and *'generalized life'* are introduced with additional constraints regarding physical coherence and logical consistency.

Here, the definition of 'life' of an artifact formalizes part of the intuition regarding the 'behavioral space' of artifacts (e.g. building element), discussed earlier at the end of Chapter III about IFC (see subsection 3.4.5). More specifically, it allows to describe all relevant processes and events an artifact participates throughout its life, in terms of a mereological fusion of perdurants. This provides a basis for the representation of behavioral effects from the supply side of functions, including side-effects with potential functional roles. A similar approach could be applied for the representation from the demand side, concerned with the specification of required functions. Indeed, the principle of parthood between perdurants,

allows the description of functions at multiple levels of abstraction, providing a mechanism to reconcile different types of function by means of mereological aggregations and overlaps of different perdurants.

As mentioned before, the criteria of selectivity and intentionality of an agent regarding the nature of these behaviors is not needed, in principle at least, to apply the principle of co-participation of two different endurants in the same perdurant.

However, selectivity and intentionality become important when the elucidation of multiple aspect systems is necessary, in order to identify potential conflicts across different functional requirements. For that purpose, functional intent and expectations of performance associated to different stakeholder viewpoints need to be made explicit.

The meaning of function in Chandrasekaran and Josephson, as *selected and intended output behavior* provides the specification required to convey selectivity and intentionality that differentiates functions from other output behaviors. Formally, an output behavior that is selected and intended by an agent G is defined in DOLCE-FR as:

$$OutBeh_{\mathrm{G}}(a, e, b) := Beh(a, e, b) \wedge OutPD_{\mathrm{G}}(a, e) \tag{18}$$

Where $OutPD_{\mathrm{G}}(a, e)$ is an output perdurant that is believed to be possible, and it is intended by an agent G as effect (possibly constrained) to play a functional role, i.e. contributing to the satisfaction of a functional requirement. For that to happen, the agent has to believe that e is a possible perdurant, otherwise its functional implications are meaningless. Similarly, a definition for input behavior ($InBeh_{\mathrm{G}}$) with regard a possible input perdurant is provided in [27].

In other words, the notion of input/output behaviors introduced in the original FR scheme, are now treated as qualities describing how an object is intended to participate in two very specific pairs of perdurants that make-up its life (i.e. the fusion of all perdurants that an object participates). In particular, the pair of perdurants are part of the life of an object, selected by designers as intended means of interaction with the outer environment.

Table 8: Behaviors of thermostat and components. Adapted from [27].

1	General behavior of thermostat	*Bringing* furnace ignition	$Beh_{\mathrm{G}}(thermostat, e, b)$
1.1	Temperature drops at $17C$		
1.2	Strip bends to angle A		
1.3	Switch closes		$Beh_{\mathrm{G}}(thermostat, e', b')$
1.4	Current flows to furnace		
1.4	Furnace ignites		
2	Behavior of bimetal strip	*Bends* to angle A	
3	Behavior of electric switch	*Closes* electrical circuit	$Beh_{\mathrm{G}}(switch, e', b'')$
4	Behavior of electric conductor	*Transmits* electrical current	
5	Behavior of furnace	*Heating* water	

5.1.7.1 Two examples of formalized behaviors

At a more general level, when intentionality of design is not the focus of the description, the less constrained formalization of behaviors suffice. DOLCE-FR provides some examples of these. Here only two are presented for convenience. First, the behavior of a lintel, originally from Chandrasekaran and Josephson[76]:

$$Beh(lintel, e, b) \wedge PD_{\mathrm{G}}(e) \wedge ST(e) \wedge P(e, Alf(lintel)) \tag{19}$$

This describes under the DOLCE-FR formalization, the behavior of a lintel distributing the vertical loads. Implicit in this formalization are other state variables, including structural relations, that are required in this situation. Since the behavior b is a quality (e.g. instantiated property) of the lintel, and the lintel, a component of a door or window frame, these other state variables can be derived. Through b, the participation of the lintel in the perdurant of load distribution gets qualified. Notice that e is a *stative* perdurant. It is also only one of many other possible parts of the life of the lintel. For instance, in the case of a door, it also participates in the processes of circulation of people and goods through the door, based on the clearance it affords.

The second example related to the behavior of a thermostat is introduced to illustrate additional advantages of the formalization. Here, the behavior refers to the participation of the thermostat in an ordered sequence of events. Altogether, these events indicate part of the actual life of the thermostat, thus narrowly defined.

Notice that the (output) behavior of the switch (row 3) describes how the switch participates in the same perdurant e' in which the thermostat participates (row 1.3). In the case of the thermostat, the type of participation is not only different, but also at a higher level of abstraction, as consequence of state variables and relationships implied by the uniqueness of b'. The participation of the switch is qualified differently by b''.

5.1.8 Behavior environment and mode of deployment

The notion of function as behavioral constraint desired by an agent introduced in FR approximates the notion of *performance requirement* discussed by Augenbroe in [16]. Such a behavioral constraint may be unconditional (e.g. an output value lower or higher than x), or conditional (e.g. if input value y, then output value x). In these cases, not only an output perdurant is intended, but that the some property value of the output needs to be met. This constraint is specified by the behavior descriptor, i.e. the specification of how an artifact is intended to participate in a perdurant, so that a property of this perdurant satisfies the constraint. In the example above, the temperature in which the thermostat is set to ignite the furnace, the temperature at which the furnace needs to heat the water, or how fast the water needs to get heated, are examples of such behavioral / performance constraints. In general, the more demanding the performance constraint, the more complex the aspect system.

The notion of *behavior environment* is formalized in DOLCE-FR to address the specification of functions as intended behavioral (performance) constraints that entities of the subclass X of technical artifacts $TechArt$ need to satisfy. Broadly speaking, a behavior environment a is the fusion of all entities of X, for which behavioral interactions need to be constrained. This is formally defined according to:

$$BehEnv(a) := \exists X(X \subseteq TechArt \wedge a = \sigma x(x \in X)) \tag{20}$$

Then, a *behavioral constraint* in environment a is defined as:

$$CrBeh(a, b_0, b_1) := BehEnv(a) \wedge (b_0 = b_1 \rightarrow \exists a'(P(a', a) \wedge Beh(a', b_0))) \wedge$$
$$(b_0 \neq b_1 \rightarrow \exists a', a''(P(a', a) \wedge P(a'', a) \wedge Cond(b_0, b_1) \wedge$$
$$Beh(a', b_0) \wedge Beh(a'', b_1))) \quad (21)$$

The intuition behind this formalization of a behavioral constraint is that behavior b_0 is a (pre-) condition for behavior b_1 under a behavior environment a. This conditionality is expressed by $Cond(b_0, b_1)$. It applies in cases where behavioral constraints are stated in conditional terms (e.g. if input value y, then output value x). For cases where behavioral constraints are unconditional b_0 and b_1 are equal.

A performance requirement in [16]entails an environment-centric specification of such conditionality, given that some 'things' need to happen in the world in certain ways, in order for other 'things' to happen in the expected way. For instance, temperature and relative humidity within a given range are conditions for thermal comfort. This in turn could be a condition for certain expectation of productivity in an office environment, etc. Different parts of the environment participate in different ways in the accomplishment of these conditions. However, the dimensions of causality and intentionality of an agent G entailed by the specification of a performance requirement require additional formal characterization. In DOLCE-FR this is defined by the notion of a *desired* behavioral constraint, using the predicate $DES_G(a, b_0, b_1)$. This is defined as:

$$DES_G(a, b_0, b_1) := CrBeh(a, b_0, b_1) \wedge \exists a', a''(P(a', a) \wedge$$
$$P(a'', a) \wedge \forall e_0, e_1((b_0 \neq b_1 \wedge Beh(a', e_0, b_0) \wedge Beh(a'', e_1, b_1))$$
$$\rightarrow INT_G(e_0) \wedge INT_G(e_1)) \wedge \forall e_0(b_0 = b_1 \wedge Beh(a', e, b_0))$$
$$\rightarrow Beh_G\,(a', e, b_0)) \quad (22)$$

These two axioms lead to the definition of the notion of device-centric function, presented below. The definition of the term $SatCrBeh(a, b_0, b_1)$ is not presented here for brevity, but

it is a further specification of the conditions, or intended states of the world, required for a behavioral constraint $CrBeh(a, b_0, b_1)$ to be satisfied.

$$DevFunc_G(a, b_0, b_1) := SatCrBeh(a, b_0, b_1) \wedge DES_G(a, b_0, b_1) \tag{23}$$

A consequence of this definition is that the function of a technical artifact is a behavioral constraint imposed in an environment, which coincides with the artifact ([27], pg. 17). This feature will be implemented in the proposed framework and demonstrated later in case study 7.2.

It is clear from the criteria of uniqueness introduced by axioms (11 to 14), that for each object participating in the same perdurant e, a different behavior exists. Conversely, the formalization of behavioral constraint in (21) implies that for each pair of conditional behaviors, an unique pair of perdurant exists. Therefore, given b_1 as a post-condition of a behavioral constraint, then an output behavior $OutBeh_G(a, e'', b_1)$ exists with an unique output perdurant e'' from (18). The same applies to b_0 as pre-condition of a behavioral constraint, for which an input behavior $IntBeh_G(a, e', b_1)$ exists with an unique perdurant e', unless b_0 and b_1 are the same behaviors.

This is relevant because e' and e'' are part of the life of the artifact or system for which the behavioral constraint applies. This means that both need to stand in some causal relationship, for which DOLCE-FR provides formalization. however, since causal relations may exist that are not intentional, further constraints allow to make the difference clear with respect a *mode of deployment* underpinning the formulation of an *environment-centric function*. According to Chandrasekaran and Josephson, a mode of deployment for an artifact a consists of the specification of (i) the structural relations between a and other objects of the same environment; and (ii) the actions between a and these objects. In DOLCE-FR, a *mode of deployment* is represented based on the notion of perdurant, and defined according to:

$$MD(e, a, a') := TechArt(a) \wedge BehEnv(a') \wedge P(a, a') \wedge$$
$$\exists a_1 (P(a_1, a') \wedge a \neq a1 \wedge PC_{\mathrm{WH}}(a, e) \wedge PC_{\mathrm{WH}}(a_1, e)) \quad (24)$$

Where $TechArt$ is a technical artifact, $BehEnv(a')$ is an environment composed by all the artifacts that define a system (e.g. an office space, a classroom, a mechanical system, an entire building, etc.). The mereological parthood relation $P(a, a')$ means that a is also part of the behavior environment a'. Additionally, there is another artifact a_1, which is also part of the behavior environment a', so that both a and a_1 wholly participate in perdurant e. Because of the constraints defined in (11) to (14), each participation relation must be qualified by a different behavior quality, so that we have $Beh(a, e, b')$ and $Beh(a_1, e, b'')$, where $b' \neq b''$.

In other words, a mode of deployment of a technical artifact a in a behavioral environment a' is a **perdurant** in which both a and a' wholly participate. This characterization allows to capture indirectly the structural relationship between a and other elements in the environment, as well as the interactions among them, via a common perdurant in which they participate.

According to Borgo, by changing the perdurant, a different set of entities, interactions and relationships gets selected (in [27], pg. 17). This observation provides the basis for the hypothesis of this research, according to the criteria introduced in the preliminary hypothesis.

To understand its relevance, is necessary to clarify the meaning of the relation between a mode of deployment $MD(e, a, a')$, and an environment-centric function. In DOLCE-FR, that relation does not mean that an artifact a in a mode of deployment e cause the function to be satisfied. Instead, it is the mode of deployment e, in which the artifact a participates, that *causes* the function to be satisfied. In other words, it is a perdurant, or more precisely, a combination of perdurants that compose a mode of deployment, which causes a function to be satisfied, and not artifacts by themselves.

For that purpose, the notion of causality has been defined in DOLCE-FR according to:

$$Cause_{MD}(a, e, b_0, b_1) := SatCrBeh(a, b_0, b_1) \wedge$$
$$\forall a_0, e_0(Beh(a_0, e_0, b_0) \rightarrow Cause(e, e_0)) \wedge$$
$$\forall a_1, e_1(Beh(a_1, e_1, b_1) \rightarrow Cause(e, e_1)) \quad (25)$$

Auxiliary definitions have been omitted here. Nevertheless, the main point is that $Cause_MD(a, e, b_0, b_1)$ established general conditions of causality between a mode of deployment e and some intended perdurants (e.g. actions and effects) specified by the behavioral constraints b_0 and b_1.

Finally, a formalization of the notion of environment-centric function is provided. Notice that $FMD(e, a,')$ is a specialization of mode of deployment, admitting only engineering perdurants that are feasible:

$$EnvFunc(b_0, b_1, a, a', e) := Cause_{MD}(a, e, b_0, b_1) \wedge$$
$$FMD(e, a, a') \wedge \exists DES_G(a', b_0, b_1) \quad (26)$$

The interpretation of which, according to Borgo et al. (in [27], pg. 18) is:

> "A behavioral constraint is said to be a function of a technical artifact in a certain environment relative to a mode of deployment e if e causes the behavioral constraint and there is an agent G that desires this behavioral constraint."

The use of the term 'technical artifact' is a reference to single device or system playing the major functional role (i.e. functional participation). Other parts of the environment playing secondary roles however are implicit in the definition. To clarify, recall in (24) that artifact a_1 also participates in the mode of deployment e, but for which there is no explicit account. It is the aggregation of this and other participants that constitute the aspect system of e.

The specific problem addressed in this research is to support the incremental identification of such sets of elements, as they are created and continuously modified during the

design process by various stakeholders and decision-makers. In particular, elements with functional participation are to be identified and harvested from BIM models, which provide a description of the technical environment for relevant modes of deployment.

The dependency among entities that conform a mode of deployment, explained in the previous section, approximates the concept of an aspect system. By changing the behavioral constraints specified for a technical artifact, the set of elements co-participating also changes, as result of new interactions. However, there is no explicit formalism to capture the changing structure of co-participation. This is relevant in the context of BIM workflows, because it is precisely the aggregation of all of these elements, under different functional viewpoints, what constitutes the information content required to support various design tasks involving interdisciplinary collaboration. In particular, this is relevant in the context of performance-based design, which currently lacks computational support for more systematic multi-criteria evaluation and decision-making.

Such cross-cutting, view-dependent aggregations denoting the aspect system of a function cannot be captured by single composition hierarchies used in traditional taxonomies of building systems, such as those provided in IFC or proprietary building data models. Instead, their elucidation requires inference capabilities over behavioral interactions taking place under particular functional contexts.

In order to support automatic elucidation of aspect systems, it is necessary to provide a formal characterization of the multiple meanings of function and behavior that exist in the AEC domain. This involves two main issues. First, the characterization needs to reconcile both the 'demand' and the 'supply' side of functions, so that incremental mappings between design requirements and candidate solutions could be better supported through modeling of relevant behavioral effects. Second, such characterization needs to enable the description of functionality at multiple levels of abstraction, making explicit causal relationships and interdependencies between low level and high level functions. This is especially important for the design of socio-technical systems such as buildings, where the value of preferred outcomes is usually associated to performance of high-level functions, such as safety, physical and psychological comfort among other relevant human factors.

5.2 Hypothesis

For that purpose, this research proposes that a new ontological commitment is necessary for the development of a functional modeling framework capable of addressing these two issues in a comprehensive, operational way. In particular, the upper-level category of perdurants, discussed in relation to the 'Aristotelian Sextet' (presented in figure 5.4), provides an additional level of representational expressivity to describe the type of semantic relationships required. These include general relationships such as participation of structural entities in perdurants, parthood between perdurants and causality between perdurants.

However, a higher degree of specificity is needed, which is given by the formalization of the Functional Representation schema under DOLCE, called from now on as DOLCE-FR. Within this ontological framework, the hypothesis of this research is that aspect systems can be formally represented in an operational, machine-readable manner, based on the concept of *mode of deployment*, and its formalization in terms of perdurants, as proposed by DOLCE-FR (see definition 24. This leads to the reformulation of the two criteria introduced in the preliminary hypothesis:

1. Co-participation of two or more *physical endurants*, possibly from different building sub-systems, and from different levels of structural composition, in any part of the same intended perdurant specified as *mode of deployment* e, independently of the intention of a stakeholder G regarding the co-participation, is a necessary condition to include such *endurants* as part of the aspect system of e.

2. Participation of the same *physical endurant* in any part of multiple intended perdurants, each specified by an unique *mode of deployment* $\{e_1, e_2, \ldots, e_n\}$, and independently of the intention of a stakeholder G regarding the multiple participation, is a necessary condition to include such physical endurant as part of all aspect systems corresponding to each mode of deployment $\{e_1, e_2, \ldots, e_n\}$.

To test the hypothesis, a proof of concept for a functional modeling framework has been developed, based on the translation of a subset of the DOLCE-FR axioms from first-order logic into description logic using the Web Ontology Language OWL-DL. This allows to

leverage the inference capabilities of available OWL-DL reasoners to support the identification of co-participation relations required for incremental elucidation of aspect systems. Inference of co-participation is based in part on information asserted during the course of a design process in the form of CAD / BIM models. This involves the behavioral characterization of design functions from the 'supply' side, that is, the behavioral space of building systems and components. It also involves the behavioral characterization of design functions from the 'demand' side, concerned with the specification of intended behavioral effects, i.e. required functions, as well as behavioral constraints, i.e. performance requirements.

Figure 5.2 illustrates the hypothesis. Artifacts a_1 and a_2 are asserted as having main participation in their corresponding modes of deployment, depicted by ovals e_1 and e_2. There is another mode of deployment e, which includes both e_1 and e_2 as its parts. The artifact a_1 has an internal component $a_{1.n}$, which has an implicit co-participation in e_2, by means of the side-effect denoted by behavior b'. According to definition $O(x, y)$ in (2), this co-participation implies a mereological overlap between e_1 and e_2.

The diagram illustrates a pattern that generalizes common issues of systems integration in Building Design, such as heat gains, noise, vibration and other types of energy or mass transfer resulting from internal side-effects of different building subsystems. It is important to indicate that these side-effects and their interactions are not limited to the category of artificial systems, i.e. artifacts. Certainly, occupant behavior and other human activities are also a fundamental concern for performance-based systems integration usually associated with high-level building functions [177, 120, 4]. Furthermore, certain animal activities or biological processes may also produce side-effects which can be captured by general patterns of behavioral interaction. For instance, aerosols such as mist droplets produced as side-effect of cooling towers may be a potential source of Legionella bacterium (i.e. Legionnaire's disease).

To respond to the level of generality required to cover different meanings of function and types of behavioral interaction in buildings, the domain of participation have been extended from technical artifacts (i.e. $TechArt$) to the broader category of physical endurants PED. This allows to describe objects that are not technical, yet may play a functional role (e.g.

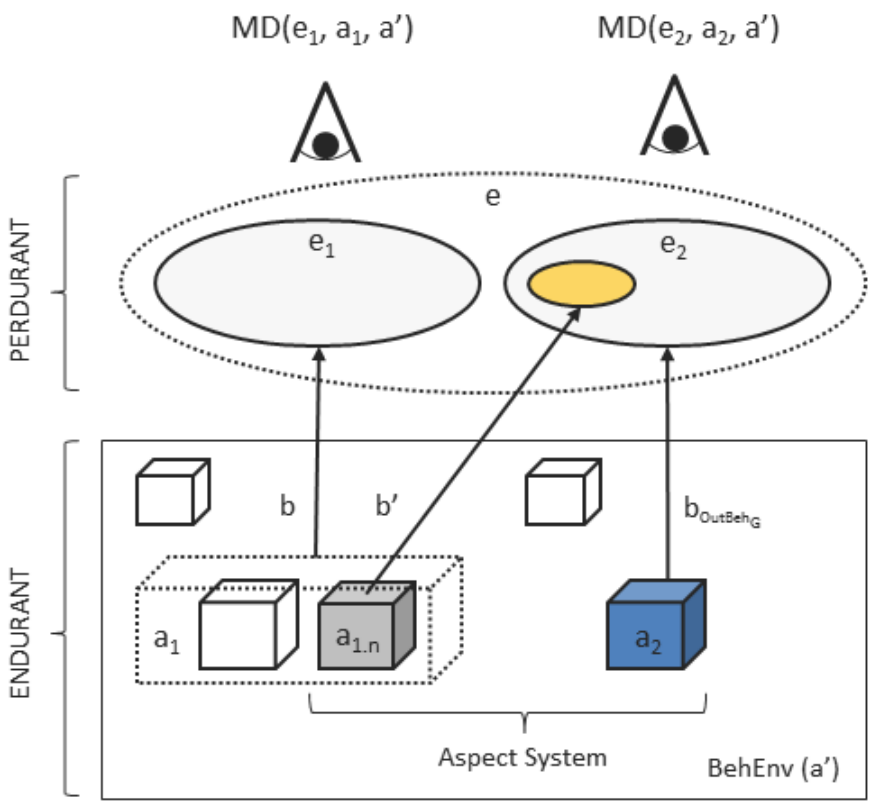

Figure 5.6: Functional integration and multi-functional criteria. Aspect system inference due to co-participation through parthood transitivity.

trees, animals, natural elements from the environment, etc.).

The condition that endurants may participate in any part of a same intended perdurant e is proposed here to relax the constraints originally imposed in the definition of a mode of deployment regarding *wholly participation*. In particular, such constraint suggests the need for complete prior knowledge regarding wholly participation of instances in the phenomena of interest. Instead, wholly participation may be known for parts of e, which allows inference of participation of elements at different levels of abstraction. This is due to the transitivity property of parthood relations that apply for both endurants and perdurants. Other inference rules involving transitivity may apply regarding causality, which can be described at a general level in terms of causal preconditions and post-conditions.

The validation of the hypothesis involves testing whether the formalization of aspect systems based on DOLCE-FR, and more specifically, based on the definition of mode of

deployment as mereological fusion of perdurants, supports the inference of functional co-participation. Underpinning this process, other inferences are needed regarding relations of subsumption, parthood and causality at multiple levels of abstraction. Principles of inheritance and transitivity over perdurants can provide a formal mechanism to support these types of inferences, which can also be the target of more specific queries.

At the most general level, the validation through a proof-of-concept involves two basic queries, which do not require specification of intentionality of agents. This approach would allow to approximate the meaning of mode of deployment, while keeping the implementation of the prototype simple. For a similar reason, the initial meaning of function to be adopted is that of *intended effect*, without further qualification of behavioral constraints (i.e. performance requirements). This allows to simplify the specification of a required function in terms of an intended perdurant e.

Thus, the first query addresses the inference of individual aspect systems, according to the first criterion of co-participation of physical endurants (e.g. building elements) in the same intended perdurant, i.e. a required function:

- Given an intended perdurant e (i.e. the required function), in a behavior environment a', return the set of building elements with participation in e.

In this query, the returned set of building elements represents the aspect system of function e. The criterion for this query has been discussed in the preliminary hypothesis as the functional integration criterion.

The second query addresses the inference of interactions and potential conflicts between different functions, according to the second criterion of participation of a same physical endurant in multiple intended perdurants, i.e. multiple required functions:

- Given a building element a, in environment a', return the set of intended perdurants $\{e_1, e_2, \ldots, e_n\}$, in which element a participates.

Implicit in the return set of this query, are the various aspect systems that may contain element a as their part. In practical terms that could be considered as an equivalent for of

query. However, aspect systems belong to the category of physical endurants, whereas the query above specifically returns a set of perdurants denoting functions. This aspect been discussed in the preliminary hypothesis as the multi-functional criterion, which is concerned with identification of all the functional roles played by a single building element. Notice that these functional roles can take place at a different levels of abstraction, ranging from local and device-centric, to more global, environment-centric functions. The latter can include 'soft' functions that are not necessarily technical, or amenable to quantification, insofar they can be specified in terms of perdurants.

These two queries represent the most general form, for which the participation relation is not qualified in terms of intentionality or behavioral constraints, i.e. performance requirements. The ability of a proof-of-concept to provide answers that are complete, sound and consistent relative to an initial knowledge-base, is considered sufficient for the validation of the hypothesis. Therefore, these two general queries define the scope of the implementation effort, and consequently, the scope of the research as a whole. Future work will need to address more specialized forms of query dealing with characterization of intentionality and specification of behavioral constraints necessary for the development of useful applications.

5.3 Methodology

The methodology for hypothesis validation involves the implementation of a proof-of-concept of a functional modeling framework capable of answering the queries outlined above. This implementation considers two major steps. The first one is the formalization of the meaning of aspect systems according to the definition of mode of deployment developed in DOLCE-FR.

The second step involves the translation of this formalization, plus a subset of supporting DOLCE-FR axioms from first-order logic (FOL) into description logic using OWL-DL. Once the translation is done in the form of an ontology model, a knowledge base model will be specified according to this ontology to capture domain-specific information regarding Building Design. This will allow the formulation of inference rules and queries, to be applied over instance models of buildings or building subsystems. Ideally, these instance models

should be provided directly by BIM applications, from which a series of design properties and relationships would be already asserted. However, only a logic representation of such instance models will be used for the validation, without explicit geometric information. These models will be based on a series of cases studies, following some of the real-world examples already discussed previously.

In summary, the implementation of the proposed framework considers four different models, according to the list below:

- Ontology model: *Translation of DOLCE-FR definitions from FOL into OWL-DL.*
- Knowledge-based model: *Generic interaction patterns.*
- Structural model: *Building systems as OWL-DL individuals.*
- Query / Rule model: *Initial SWRL rule expressions and DL query expressions.*

Consistency checking: *Check semantic validity of query / rules results.*

The third and fourth steps support the validation of hypothesis, through the development of case studies based on ontology model. Examples of aspect systems discussed previously will be used for reference. The specific steps for validation involve the development of instance models with initially asserted relations. These include (i) structural relations of connectivity and parthood in the behavior environment, (ii) combinations of behavioral properties, (iii) combinations of participation relations.

Mention / list the three cases studies.

5.4 Scope, contribution and possible impact of the research

The formulation of the hypothesis is motivated by the current lack of tools and methods to support the interdisciplinary and collaborative elucidation of subsets of a design model that are meaningful from various functional perspectives. These subsets define the operational context or *mode of deployment* for each functional viewpoint, and from which multi-criteria evaluation of performance needs to be made.

Given the degree of modeling flexibility implied; the solution needs to be different from other approaches that rely on the prescription of model views (e.g. IFC Model View Definitions) intended to fulfill normative sets of data exchange scenarios [176].

The requirement of representational flexibility needs to be satisfied first by the addition of an extra layer of semantics, on top of current data models, so that the characterization and processing of design geometry can be performed from different functional perspectives. Another aspect in which representational flexibility needs to be provided is in the dynamic identification and assignment of functional relations as the design model evolves. This capability demands a degree of automatic reasoning in order to infer when new relationships and cross-cutting functional inter-dependencies under conditions of partial knowledge and uncertainty.

Altogether, semantically enhanced representations and reasoning capabilities can contribute to the development of various design activities, from specification of functional goals and performance requirements, generation and modification of designs, evaluation and comparison of design alternatives, validation and verification of performance, and multi-stakeholder negotiation and decision-making. To provide a shared understanding of the multiple meanings of function is one of the most important objectives of a functional modeling framework in Building Design.

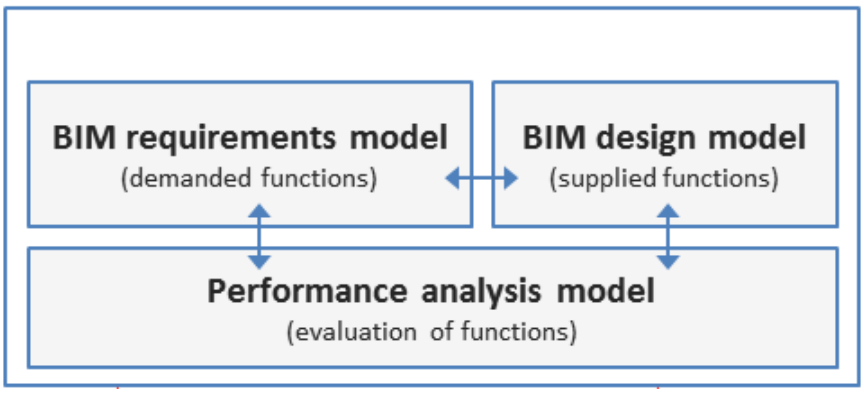

Figure 5.7: Integrated functional modeling framework.

These requirements for functional modeling framework define the scope of this research, which can be summarized in terms of the different functional viewpoints that need to be

described, either by means of explicit assertions made in a design model, or derived by means of inference. The following list provides an overview of these functional viewpoints, which together, define the scope of this research.

- **Supply and demand:** A common representation of functions that serve both, the specification of required functions, and the specification of functions provided.
- **Multiple levels of abstraction:** Domain-specific functional viewpoints (horizontal integration) and general functional viewpoints (vertical integration).
  - **Horizontal integration (domain-specific functional meaning):** Integration of domain-specific meaning of functions associated to different technical sub-systems at the same level of aggregation.
  - **Vertical integration (hard and soft functions):** Integration of functional meaning across different levels of abstraction. This involves the problem of how the semantic content of low-level functions, usually technical and quantifiable (e.g. heat pumps), relate with the semantics of intermediate and high-level functions, which usually are not technical or easily quantifiable (e.g. productivity).
- **Distal functional dependencies:** Two additional functional perspectives required have to deal with how entities that are distant from each other in space or time are described as functionally related.
- **Nominal and anonymous functional roles:** This class of functional viewpoints relate to the fact that many parts of buildings perform more than one main nominal function. At the same time, many functions are affected by the participation of multiple parts, either negatively or positively. The set of these parts is the aspect system of the function, and the specific goal of this research is precisely to enable the identification of these parts.

5.4.1 Generalization of research findings

The ontological commitment in design representations to the category of perdurants, and to the new relational category of *participation* between endurants and perdurants provide

a formal and richer extension to the semantics of current design data models. While this research focuses exclusively on the formal description of the class of engineering perdurants required to support design-analysis integration in performance-based design, the framework could be extended to cover other kinds of perdurants relevant to other forms of design and design activities.

It is anticipated that this semantic extension could support more systematic methods for the identification of meaningful subsets of building models, according to the various perspectives in which time-dependent phenomena, actions, events, states and processes need to be taken carefully in consideration. These perspectives or aspects include, but are not limited to: design for manufacturing and a set of common manufacturing requirements and processes, assembly, constructability, systems operations and maintenance, malfunctioning diagnosis and retrofitting, etc.

As mentioned before, this approach is intended to be more flexible than a normative prescription of building model subsets, as strategy to satisfy predefined data exchange scenarios. However, it is most likely that both approaches would be complementary to each other. In particular, a predefined model view could be used as a standard, more complete baseline of input assertions from which better and more complete model inferences can be derived.

With current efforts in the unification of the meaning of function across various disciplines, including biology, medicine and social sciences, to name a few, eventually other types of functional aspects related not only to buildings, but to the entire built environment could developed. On the other hand, there are efforts in the IFC community to make use of semantic web technologies to improve distributed data exchange workflows based on OWL-DL. The work proposed here could contribute to that effort by introducing an much needed additional layer of semantics.

CHAPTER VI

FORMALIZATION OF ASPECT SYSTEM

In this chapter the hypothesis of the research is investigated, in terms of formalizing the meaning of the concept of aspect systems within the formalization of DOLCE-FR. Then, the proposed formalization along with a required subset of DOLCE-FR is implemented in OWL-DL, as part of the experiment model proposed for the validation of the hypothesis. Issues of implementation will be discussed, in relation to the process required for translation of FOL into OWL-DL, expressivity and complexity of the model and computational cost involved in reasoning and querying. Finally, three case studies will be presented for final validation based on a series of queries that capture the criteria established by the hypothesis.

6.1 Formalization of aspect systems

An explicit description of the functional environment where effects and interactions of interest take place is crucial for elucidation of aspect systems. In FR, this description corresponds to the so-called *mode of deployment*. A mode of deployment involves a dual specification of structural and causal relations that need to hold between a device and other entities of the environment. In DOLCE-FR, this dual specification is formalized under a more strict ontological distinction, grounded on the DOLCE categories of *endurant* and *perdurant*.

In this formalization, the specification of structural relations is made with the use of the notion of *behavior environment*, represented by the predicate $BehEnv(a)$, defined in (20). Intuitively, a behavior environment is the fusion of all structural entities (e.g. the entire physical system) on which a *behavioral constraint* applies. Recall from FR that the notion of behavioral constraint is one of the meanings of function usually found in engineering practice. It seems clear, at a general level at least, that this is the case for the meaning intended in the characterization of performance requirement proposed Augenbroe, and discussed in Chapter 1.

Regarding the specification of causal interactions, this gets formalized by the mode of

deployment itself, represented by the predicate $MD(e, a, a')$, defined in (24). A mode of deployment is characterized as a single *perdurant*, or more specifically, as a single *fusion of perdurants* e, in which both a technical artifact a and an environment a' wholly participate. Thus, the formalization of a mode of deployment is an abstraction combining the description of an overall effect, i.e. the perdurant e, and participants of these effects i.e. the endurants a and a'. The perdurant e can then be used as index of the abstraction.

This formalization of mode of deployment has a number of important implications. In particular, that a change in the specification of the perdurant e in $MD(e, a, a')$ implies a change in all entities, interactions and relationships involved. In this formulation however, changes are not explicitly linked to causality and intention of stakeholders. Further definitions in DOLCE-FR introduce that type of semantics required for the final formalization of environment-centric function $EnvFunc(b_0, b_1, a, a', e)$.

For practical purposes however, the dependency among entities in $MD(e, a, a')$ remains relevant. The assumption of the research hypothesis is that those additional semantic constraints can be skipped for the sake of simplicity. For this reason, the focus in on a minimum set of axioms considered sufficient to meet the two criteria discussed in hypothesis presented in (5.2).

Given the condition of dependency among terms of a *mode of deployment* $MD(e, a, a')$, it is clear that its meaning approximates the meaning of *aspect system*. To Illustrate this point, it is necessary to consider entity a not as a single, discrete technical artifact, but a more generic aggregation set of physical endurants, then the following observations can be made regarding a mode of deployment $MD(e, a, a')$:

(i) The perdurant e specifies the effect on the environment that is intended by design. This can be further specified with behavioral constraints akin to the notion of performance requirement.

(ii) The endurant a is an aggregation of parts of the environment a' having a participation in e.

(iii) Not all parts of environment a' participate in e.

(iv) Changes on the specification of e or a' modify the structure of participation of a', and by consequence, the membership of a itself.

Remember that the relation between a mode of deployment $MD(e, a, a')$, and an environment-centric function $EnvFunc(b_0, b_1, a, a', e)$ has a very particular meaning. In DOLCE-FR, this relation ***does not mean***, as it is usually assumed, that an artifact in a certain mode of deployment causes a function to be satisfied. Instead, what it actually means is that the mode of deployment is a perdurant, in which the artifact participates, and that *it is the mode of deployment* what that causes a function to be satisfied. This implies the participation of other artifacts in a', including parts of a itself.

A general notion of causality is defined in DOLCE-FR with the formula $Cause_{MD}(a, e, b_0, b_1)$, where a is the environment, e is the perdurant indexing a particular mode of deployment (denoted by the subscript in $Cause_{MD}$), and b_0 and b_1 are the behavioral constraints for which causality needs to hold. More specifically, mode of deployment e causes the behavioral constraint to be satisfied. The details will be discussed later when the implementation is presented.

However, besides $TechArt(a)$, which is typically allocated the 'nominal' environment-centric function of interest, DOLCE-FR does not seem provide a method to refer explicitly to other parts of the environment a' that also may participate on e. Considering the modeling requirements for an aspect system, other elements playing a secondary role, or for which the participation status may change overtime, also require formal description. The requirement of incremental elucidation applies when design elements are introduced, modified, or removed from the overall system in response to additional behavioral constraints specified elsewhere.

It is easy to see how this takes place in practice. For instance, the more demanding the behavioral constraints related to say, safety in a building, the higher the diversity of events and activities (i.e. use case scenarios) that need to be part of the mode of deployment. Entities that previously were not considered as playing any role might acquire a different status under new circumstances. Consequently, a different set of participating objects, features and technologies need to be included in the aspect system of the more stringent 'safety'

requirement. Conversely, objects previously considered 'harmless' cannot be brought inside a building, which in turn may affect the performance of other functions, under different modes of deployment.

For instance, the mode of deployment for thermal comfort is a fusion of perdurants involving various processes of heat and mass transfer. In building design, these type of environment-centric functions are typically allocated to individual technical systems, such as HVAC systems. This was discussed in Chapter 3, were it was noted that most BIM schemas provide standard representations for conventional technical systems with such main 'nominal' function. These are usually categorized according to a specific disciplinary domain, such as electrical, mechanical, structural, architectural, etc. Yet in practice, high-level environment functions such thermal comfort or safety are satisfied by a combination of co-participant entities within complex modes of deployment, requiring careful integration of various systems besides those allocated with 'nominal' functional roles.

From the example about thermal comfort introduced earlier in the section 4.3.5, it can be seen how elements as diverse as windows, HVAC systems and open, shared kitchens among others may all participate in different ways in the accomplishment of thermal comfort. Unfortunately, the required type and degree of co-participation at a high level of abstraction cannot always be asserted a priori, given the contextual nature of functional interactions. For this reason, only nominal participation of technical systems conventionally allocated the function is asserted.

Yet, there are often less obvious internal mechanisms which may cause unintended side-effects impacting high-level functionality of a system. The biological process of transpiration in plants is an example. Depending on the situation, this may have a positive functional role, whereas in other cases, the opposite may occur. The status of participation may vary in time (e.g. due to seasonal changes) or according to the viewpoint of stakeholders involved.

The definition of mode of deployment however does not allow to make explicit description of the set of elements playing secondary roles. To illustrate the point, it is necessary to consider the DOLCE-FR definition given in 24, presented below again for convenience. In this definition, the entity a is the artifact with the main participation or nominal function,

within the mode of deployment e. Entity a_1 also has a wholly participation, but besides the fact that it is also part of the environment, a_1 lacks a more explicit characterization.

$$MD(e, a, a') := TechArt(a) \ \wedge \ BehEnv(a') \ \wedge \ P(a, a') \wedge$$
$$\exists \, a_1(P(a_1, a') \ \wedge a \neq a1 \ \wedge \ PC_{WH}(a, e) \ \wedge \ PC_{WH}(a_1, e))$$

Furthermore, assertions of wholly participation are limited to the aggregate level, i.e. the fusion of perdurants e. This however requires prior knowledge regarding wholly participation of instances in the effects of interest. Instead, wholly participation may be known for parts of e and parts of environment a' that otherwise would be excluded from the description of a mode of deployment. Due to the transitivity property of parthood relations, inference of cross-cutting participation and interactions could be supported. Other inference mechanism may apply as well, based on subsumption under classes of causality, given appropriate sets of pre-condition and post-conditions.

6.1.1 Aspect systems

From the definitions provided in the formalization of DOLCE-FR, a preliminary definition for the meaning of an aspect system is formulated here using First-order logic. This formalization is based primarily in three main axioms from DOLCE-FR, which are: mode of deployment ($MD(e, a, a')$ defined in 24), causality in a mode of deployment ($Cause_{MD}(a, e, b0, b1)$ defined in 25), and environment-centric function ($EnvFunc(b_0, b_1, a, a', e)$, defined in 26). The following axiomatization is just a preliminary definition, which arguably is too general as to fulfill the specific semantic requirements regarding characterization of behavioral constraints . The intent however is to frame the rationale adopted towards an operational formalization in OWL-DL.

$$
\begin{aligned}
Aspect(s, b_0, b_1, a, a', e) \;:=\; & EnvFunc(b_0, b_1, a, a', e) \wedge \\
& PP(a, s) \;\wedge\; PP(s, a') \wedge \\
& \exists\, e_n (P(e_n, e) \;\wedge\; \forall\, a_n, e' (P(a_n, a') \wedge \\
& PC(a_n, e_n) \;\wedge\; causes(e_n, e') \wedge \\
& (beh(a, e', b_0) \;\vee\; beh(a, e', b_1)) \;\rightarrow\; P(a_n, s))) \qquad (27)
\end{aligned}
$$

Informally, this definition means that an aspect system s for the function with behavioral constraints $< b_0, b_1 >$, in mode of deployment e, aggregates parts a_n of the environment a' besides the nominal artifact a, if the following conditions apply:

1. There is a perdurant e_n, such that e_n is part of the mode of deployment e;
2. For all a_n part of environment a', and for all perdurants e' part of mode of deployment e; if it holds that:
   (a) a_n participates in e_n, and
   (b) e_n causes e', and
   (c) e' is constrained either by b_0 or b_1
3. Then a_n is a proper-part of the aspect system s

The use of some 'part' relations instead or 'proper-part' is intended to capture the fact that, in an extreme of a spectrum, a_n might be totality of the environment a' itself. Similarly, due to reflexivity of part relations, e_n might be the entirety of the mode of deployment, or just a proper-part.

The limitation of this definition however, is that the participation for a candidate member a_n in e_n, and therefore in the mode of deployment e is not qualified, as denoted by $PC(a_n, e_n)$. This is problematic because it implies that the aspect system s is essentially the same as the behavior environment a'. In other words, there is not differentiation between both classes, and therefore the definition is redundant.

A more specific characterization allowing not only to differentiate between aspect systems and behavior environments, but also to convey the intended relation of aspect systems with different constraints should be based on a qualified participation for the candidate part a_n. An approach is proposed with the next definition:

$$\begin{aligned} Aspect(s, b_0, b_1, a, a', e) \ := \ & EnvFunc(b_0, b_1, a, a', e) \ \wedge \\ & PP(a, s) \ \wedge \ PP(s, a') \ \wedge \\ & \exists \ e_n, b_n (P(e_n, e) \ \wedge \ \forall \ a_n, e' (P(a_n, a') \wedge \\ & Beh(a_n, e_n, b_n) \ \wedge \ causes(e_n, e') \ \wedge \\ & (beh(a, e', b_0) \ \vee \ Beh(a, e', b_1)) \ \rightarrow \ P(a_n, s))) \end{aligned} \tag{28}$$

The reference to the ternary relation $Beh(a, e', b_1)$ adds more specificity and control over entities to be included in the aspect system. For example, the type of behavior constraint, or behavioral capability associated to participants can be specified based on this formulation at both class and instance levels in OWL-DL. This includes the specification of input and output behaviors, along with data property values (e.g. r-values, fire ratings, acoustic absorption coefficients, etc.).

Both definitions however represent very general, broad approximations of the meaning of aspect system. More concrete definitions need to be formulated in the specification of corresponding knowledge base models. These can be based on variations of the general forms, involving constraints over different types of relations. These might include constraints on participation, causality, composition relations, and even topological and spatial relations.

The following listing illustrates one of two approaches developed for the translation of this general form into OWL-DL. This particular definition is the most recent approach developed, which relies on an equivalent class axiom for inference of participants in a given function. The other approach is based on property chains with rolification. Both approaches will be discusses in more detail in the following sections.

Listing 6.1: OWL-DL Equivalent class axiom for an aspect system.

```
Class: Aspect_system
    EquivalentTo:
      (part some (participant−in some (_c_causes some
            (inverse (_DF_outPD) some _E_function))))
      and (proper−part−of some Environment)
```

Different types of constraints, including behavioral constraints, as well as constraints on participation, causality and composition, can be added either by changing the object properties types involved (i.e. object relations), or by adding extra logical connectives. This is demonstrated in the series of axioms below, which define six variations of the general form provided by this second version of the definition of aspect system in OWL-DL

Each variation specifies different constraints on the specification of the aspect system for the environment function _fe7.2_maintain_form, which is a sub-function of the environment function of structural integrity for a photovoltaic racking system. This specific racking design is intended to operate on top of flat-roof commercial buildings. This particular type of PV racking usually relies on a combination of ballast and wind deflectors to maintain its structural form and position, without any physical connection that requires penetrations into the roof. For this reason, weight per area and aerodynamic coefficient of drag are important behavioral constraints, along with coefficient of friction of rubber pads used in the footing of the system to interface with roof membranes, whenever this condition applies.

Listing 6.2: OWL-DL Variation 1.

```
part some (participant−in some (_c_causes some Uplifting))
and (proper−part−of some Environment)
```

Listing 6.3: OWL-DL Variation 2.

```
DL query: variation 2
part some (nominal_participant_in some (_c_causes some (inverse (_DF_inPD) value _fe7.1
    _maintain_position)))
and (proper−part−of some Environment)
```

Listing 6.4: OWL-DL Variation 3.

```
part some (nominal_participant_in some (_c_causes some (inverse (_DF_outPD) value _fe7.1
    _maintain_position)))
and (proper−part−of some Environment)
```

Listing 6.5: OWL-DL Variation 4.

```
part some (participant−in some (_c_causes some (inverse (_DF_outPD) value _fe7.1
    _maintain_position)))
and (proper−part−of some PV_racking_assembly)
and ( _b_has_capability some Uplift_resistance_capability )
```

Listing 6.6: OWL-DL Variation 5.

```
part some (participant−in some(_c_causes some (inverse
behavior_constraint_on_perdurant value b_reduces_uplift_by_deflection )))
and _b_has_capability some ( has_friction_coefficient some xsd:decimal)
and (proper−part−of some PV_racking_assembly)
```

Listing 6.7: OWL-DL Variation 6.

```
Racking_component and part some (participant−in some (_c_causes some Uplifting))
and _b_has_capability value b_reduces_uplift_by_weight
or _b_has_capability some (has_aerodynamic_coefficient some xsd:decimal[<0.09])
```

The degree of specificity increases from the first version at the top, down to the last one at the bottom, where two types of behavioral constraints, specified in terms of behavioral capabilities, are used. Moreover, the second constraint is further characterized by an upper limit for a numerical value associated with the constraint (i.e. has_aerodynamic_coefficient) . The specificity also increases due to direct reference to individuals (i.e. instances) in some of the axioms, indicated by the keyword value. Other axioms rely on class references only.

The inferred members of these aspect systems are those components which have different forms of participation in the maintenance of the PV racking position (under conditions of

wind loads). Thus, the first axiom is the most general in terms of how membership to the class is specified, where any form of causal participation in the processes of uplifting may count. The only hard constraint is that the elements must be part of some environment.

Some of the participation relations are constrained either to input perdurants or output perdurants of the function. Each returns a different set of members, as it would be expected. Others introduce the notion of nominal participation as a form of constraint. This constraint captures the intuition behind the artifact a of the definitions of device-centric function, mode of development, and environment-centric function, among other auxiliary formalizations provided in DOLCE-FR.

In this framework, an artifact with nominal participation denotes a device which has been nominally allocated a function, and therefore it is assumed to have the main functional role in relation to the output perdurant of that function. Because of this, the artifact a is considered the first part of the aspect system s of the function it has been allocated. In the proof-of-concept framework, this is done by asserting the nominal role in the directly in the specification of the function itself function. It can also be done when an artifact instance is declared to be of a certain type. In this case, the nominal functional role is inherited.

Another important point to make about the different axioms presented above is related with the representational role that the notion of behavioral qualification plays in the characterization of functions. Thus, the first variation in the listing could be read intuitively as *'x has a part that causes uplifting'*, which does not really convey the meaning intended.

This sounds problematic at first glance because the goal would be to identify only those elements from the environment that do exactly the opposite, that is, to resist uplift, not those that 'cause it'. Obviously, from an engineering perspective those elements that 'cause' or 'causally contribute' to the uplifting of the PV racking are also of interest. For instance, certain corner conditions of the roof, in conjunction with parapets are known to produce turbulence, which in turn may cause the corners of a PV array to fail under a sudden increase of uplift wind loads. For this reason the corners of a PV racking may need to carry more ballast than other areas.

In any case, a finer grain distinction is needed to identify which members contribute

to the phenomena, from those that do not. Such constraint cannot be hardwired a priori, because it depends on the point of view and other conditions of the larger context. Under very different circumstances, it could be very well the case that failure induced by increased uplift forces at the corners is precisely the objective of a group of people, independently of their motivation and rationale. Therefore, the mechanism of qualification needs to be external, and to some extent arbitrary, in the sense that no high-level constraint should be imposed over how qualification is made, or which entities stand in the relation, besides the ontological constraints over the domain and range of the relation $Beh(a, e, b)$.

In the last three axioms at the bottom of the listing this ambiguity disappears, since that the participation relation is qualified by different types of behavioral descriptions. Thus, some parts participate in uplifting resisting it, as indicated by *resisting uplift*. Some others resist uplift by means of deflecting wind (i.e. wind deflectors), while other resist by shear weight (i.e. ballast units). Furthermore, resistance against uplift was further characterized by the addition of the constraint 'aerodynamic coefficient', included here for illustrative purposes.

On the same token, the lack of qualification also indicates the limitations of the interpretation made in this research about the nature of causal relations. While this interpretation has been sufficient for the modeling purposes of this prototype, future work will need to consider this from a more rigorous ontological perspective, especially regarding the nature of constraints that apply to causal relations under various conditions.

Considering that the design process of buildings is characterized by interdependent sets of functional requirements under continuous change, the difficulty of the problem of tracing relevant interactions and co-participation at multiple levels of the abstraction is certainly challenging. Any attempt to tackle this problem requires a mechanism that avoids unnecessary complexity, and the risk of combinatorial explosion that would lead towards multiple mappings of participation and causality. On the other hand, a practical ontology of functions should also avoid the risk of terminological bloat, and the proliferation of ad-hoc definitions for every variation and exception found in practice.

A possible method to reduce these risks, while ensuring extensibility and consistency

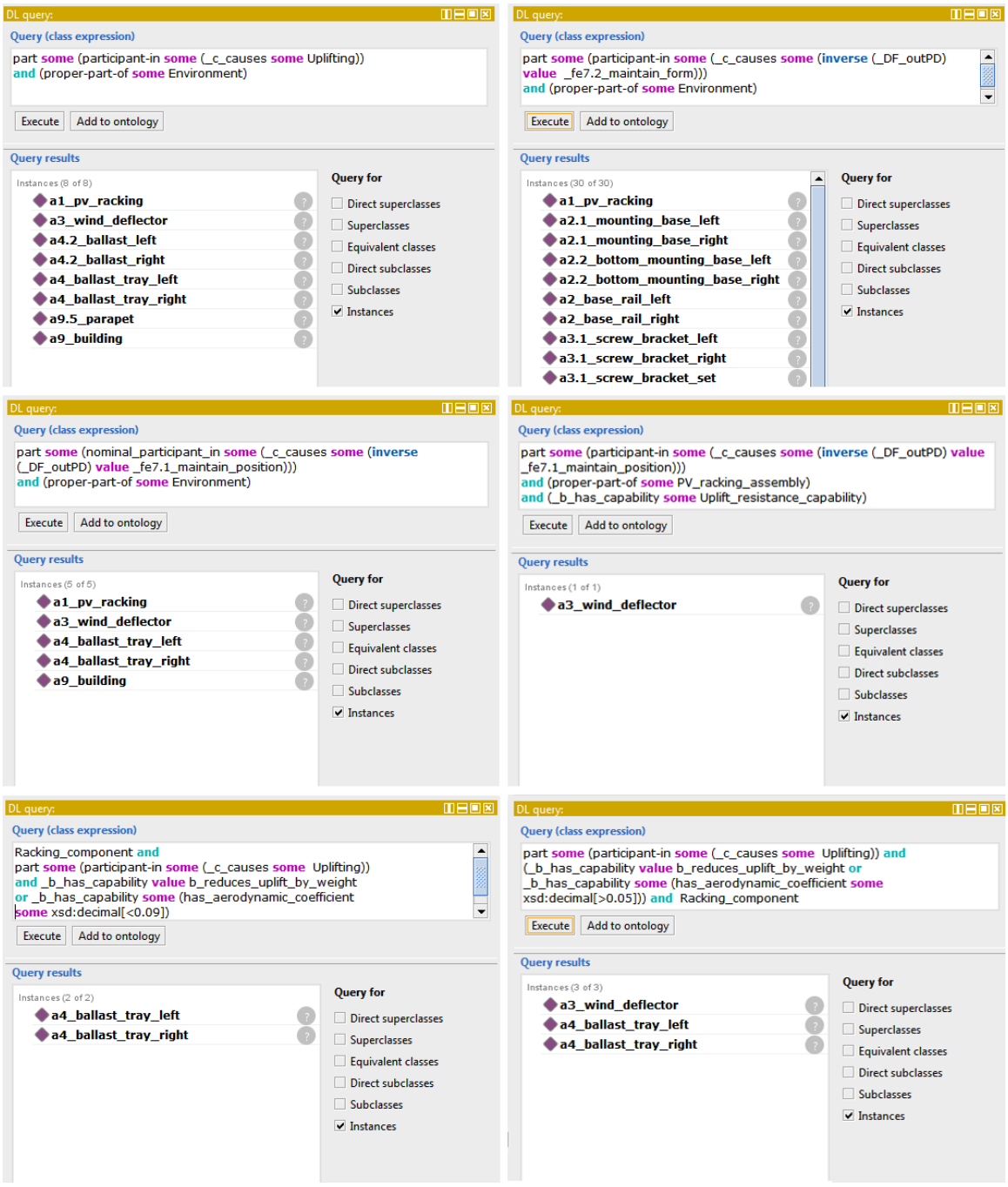

Figure 6.1: Six variations of the general form of the aspect system definition. Inferences of co-participation based on equivalent class with different sets of constraints applied.

of the model would be the systematic definition of reusable patterns, especially in the representation of causal models and classes of behavioral interaction that tend to recur in certain types of systems. In the context of buildings, many of these patterns are well known and documented, especially in the literature dedicated to the documentation of different building technologies and strategies for building systems integration.

While every building is indeed unique in terms of its location, configuration and uses, contextual conditions behind local variations can be generalized, to some extent, under stereotypical conditions. In this way, generic patterns of interaction could be formulated to cover the most common scenarios, combining instantiation of default values with different inference rules to deal with local variations, exceptions and other design conditions under partial knowledge.

Some examples of these approach have been made during the development of the case studies, with a number of environment-centric functional specifications defined by reusing and (manually) adapting some basic patterns. While the preliminary results suggest this as a viable approach, future work will be needed to validate the method under more realistic conditions.

6.2 Overview of proposed functional modeling framework

In the long term, the envisioned functional modeling framework should support the distributed nature of interdisciplinary work, to be integrated not only in CAD and BIM environments, but also in product life-cycle management (PLM) systems, and a wide variety of domain-specific applications.

An increasing number of solutions for different areas of design are becoming web-based, which promotes an even more wide-spread use over the Internet. At the same time, new interoperability challenges are likely to emerge. For that purpose, a series of research efforts are focusing on the development of technologies under the umbrella of the semantic web promoted by the World Wide Web Consortium (W3C).

Whether the semantic web project will ever get realized as originally envisioned, remains to be seen. Meanwhile, the current technologies offer a valid, practical point of reference to

situate a future functional modeling framework within a distributed architecture.

Among the technologies and standards put forward by the W3C, the Resource Description Framework (RDF) language and Web Ontology Language (OWL) are the most widely adopted. These languages provide semantic-rich, machine-readable descriptions of data, promising more advanced forms of automation and interoperability for a variety of applications. For this reason, recent research initiatives led by the buildingSMART Linked Data Working Group is developing a recommended OWL version of IFC data model, called IfcOWL [51]. The goal of this group is to leverage the potential of linked data for the generation and processing of IFC-based data across different BIM applications [250].

In this context, and considering that the envisioned functional modeling framework provides semantic layer that is independent from existing building data models, the architecture of the semantic web, and in particular the use of OWL, suggest a promising alternative. Therefore, the implementation of the proof-of-concept follows the general scheme of this architecture. At its core, there is the Resource Description Framework (RDF), a data model underlying the specification of semantic web standards for knowledge representation such as the Web Ontology Language (OWL), and the Semantic Web Rule Language (SWRL), and specification for query languages such as SPARQL. Figure 6.2 illustrates the relationship of these standards within the semantic-web stack, along with the proposed functional modeling framework.

Within this architecture, the proposed functional modeling framework consists of two main layers. The first layer (in blue, at the top of Figure 6.2) deals with the development of an integrated environment of front-end user interfaces and BIM applications. These include applications for the specification of building functions from both the demand side, as well as from the supply side, ranging from low to high levels of abstraction. From the demand side, it is necessary to enable formal, machine-readable, and model-based specification of functional and performance requirements, that is, a requirements model. From the supply side, applications should support the functional characterization of different design elements instantiated in BIM models by different BIM authoring tools. Finally, this integrated environment should also include applications for the development of analytical

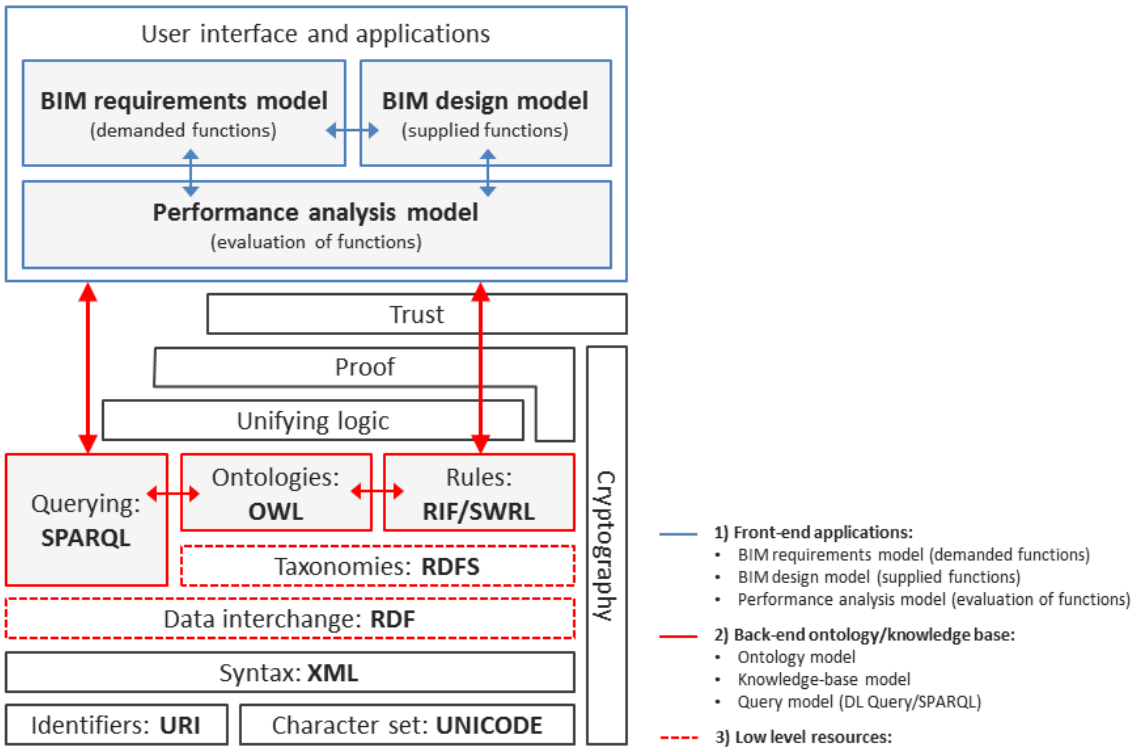

Figure 6.2: Proposed framework within the Semantic Web architecture. The Web Ontology Language (OWL), the Semantic Web Rule Language (SWRL) and SPARQL query language.

models, including different types of simulation required for performance assessment and other forms of evaluation of functionality.

Altogether, the integration of models for the specification of requirements, design alternatives and performance analysis would provide a framework for multi-criteria decision-making grounded on a common vocabulary of functions. For that purpose, an ontology model for the formalization of the demand and supply sides of functions needs to be provided, along with a knowledge base capturing the most relevant behavioral interaction patterns underlying the inference of aspect systems.

The ontology and knowledge base models are implemented in the proof-of-concept according to the formalization of DOLCE-FR. This implementation, along with the specification of query capabilities, constitute the second layer at the back-end of the proposed functional modeling framework. The definition of these back-end models rely on the adoption of OWL-DL, SWRL and SPARQL languages, depicted in red in Figure 6.2.

1. **Front-end application models**

   (a) **Functional requirements model:** This involves a formal, machine-readable description of functions from the demand side, including specification of performance requirements. It is envisioned that in the future BIM applications will support such form of specification, either internally or by integration with third-party applications sharing a common ontology of building functions. Among the main goals of a functional requirements model is to support the process of requirements specification under conditions of continuous refinement and evolution. In practice, this means the ability to support the identification, addition, modification and removal of required functionality at different levels of abstraction, along with ability to map meaningful cross-cutting interactions from different functional perspectives. This involves the description of both qualitative and quantitative behavioral constraints supporting the specification of different types of performance indicators. In the proof-of-concept, the specification of functional requirements relies on the vocabulary provided by DOLCE-FR, and implemented in OWL-DL.

   (b) **Design model:** This model is envisioned to provide a design description from the supply side of functions. This means a formal, machine-readable functional characterization of design elements instantiated in a CAD or BIM model. Currently, as discussed in Chapter 3, the conceptualization of BIM data models, such as IFC, deals primarily with the representation of structural aspects of design. This involves the description of normative object and relationship types, based on topological, geometric and composition relations that are necessary for a structural representation of buildings. Meanwhile, characterization of functional and behavioral aspects of design is treated informally, based on ad-hoc methods of property definition and association with external, classification systems that are not machine-readable. To solve this problem, a formal characterization of functions is needed, either internally, as part of the core schema of BIM models,

or externally, as an additional level of semantics to be added on top of BIM models, either at the schema of instance level. Model-based characterization of supplied functions in the latter scenario would require a series of inferences over structural information asserted in BIM models at various stages of the design process. The resulting model would provide the basic input for the elucidation of aspect systems associated to required functions specified in the requirements model. However, the full implementation of such front-end capabilities is not in the scope of this research. Therefore, the proof-of-concept presented here considers only the development of a simplified logical representation of the design model, without reference to explicit geometric information.

While a more concrete solution towards integration with BIM design models is not in the scope of this research, some observations can be made in this regard to guide future work. In general, such integration could take various forms, some of which are already under research, especially by the linked data / Ifcowl community [250]. From the specific viewpoint of this research however, the most important step would be to get reliable information about relevant structural aspects of the design. This would require a mechanism to ensure for semantic consistency with pre-validated topological and geometric relations, composition relations and properties according to element types. It is important to clarify that it is not the role of the proposed ontology or the knowledge-base models to validate the consistency of structural relations established in the design model, since it is assumed that such role is played by the source BIM or CAD application. A valid structural model of the design would already contain many assertions from which DL reasoners could generate a series of useful inferences. IN the context of an IFC model, the most important kinds of assertions would be about element types (e.g. *IfcBuildingElement* in IFC2x4), composition and aggregation relations that different building element types stand for (e.g. *IfcRelDecomposes* and *IfcRelAggregates*, along with property sets (e.g. *IfcRelDefinesbyProperties* that provide the possibility of new associative links to be generated by inference.

Many properties relevant to different life-cycle functions of buildings are systematized by classification systems such as OmniClass, which is been increasingly used for the definition of IFC property sets. A more detailed discussion of this system is provided in Appendix A.

(c) **Performance analysis model:** This model is intended to provide a formal, composite specification of both inputs and outputs associated with performance analysis applications. In particular, the objective would be to reconcile the modeling schemas of analysis applications (e.g. simulation packages) with the specification of performance requirements under the same functional conceptualization. Currently, there is no formal semantic relationship between inputs and outcomes of analysis applications with the specification of performance requirements. This is in part because of the lack of a formal model-based representation of functional requirements. On the other hand, analysis applications are usually based on very domain-specific conceptualizations of reality, concerned with the representation of physical phenomena under perspectives that are strictly disciplinary. Such limitation hinders the possibility of interoperability required to automate various forms of design-analysis integration. These include the automatic invocation of analysis routines under pre-specified conditions, as well as traceability of causal inter-dependencies impacting the satisfaction of different performance requirements. While supporting this type of automation is considered a critical capability for a functional modeling framework as envisioned here, this problem is not in the scope of this research.

The semantic integration of the three models at the front-end needs to rely on the definition of an ontology providing a compatible characterization for the different meanings of function usually adopted in Building Design. Such characterization also need to support the description of functions from the demand, supply and evaluation perspectives, possibly from different levels of abstraction. The characterization of these perspectives is further provided through the formulation of a knowledge base

model grounded on the proposed ontology of functions. This allows to make explicit domain expertise about the relationship between high-level functions (e.g. acoustic comfort), with performance evaluation criteria (e.g. reverberation time). It also allows to capture explicitly the most common patterns of behavioral interaction stemming from building systems and components with functional participation in the reverberation phenomena. This information can result from a combination of assertions and inferences triggered during the design process, and made accessible through different types of queries. The most relevant queries for the elucidation of aspect systems discussed in the hypothesis are generalized in the form of a query model. An more specific overview of the envisioned framework is provided next.

2. **Back-end ontology, knowledge base and query models**

   (a) **Ontology model:** This model translates a subset of DOLCE-FR definitions using the Description Logic variation of OWL 2 (OWL-DL). The implementation also includes the formalization of aspect systems based on the definition of mode of deployment provided in DOLCE-FR. The proposed ontology is developed with references from DOLCE-Lite, a small version of the DOLCE ontology with a rich axiomatization structure with expressivity $\mathcal{SHIQN}(\mathcal{D})$ [140]. However, only a reduced number of definitions are considered necessary for proof-of-concept validation. In particular, a minimum set of definitions and logical constraints were adopted in order to achieve the intended inference capabilities while trying to maintaining the computational cost associated with OWL-DL reasoning within acceptable limits. Specifically, main definitions regarding relationships of composition (part-of), participation and causality are used, in order to support the characterization of aspect systems according to the notions of mode of deployment and environment-centric functions provided in DOLCE-FR. Thus, the ontology model provides a shared formal vocabulary for the description of functionality from the demand, supply and evaluation perspectives. Reusable specifications of

required functions and known patterns of interaction among conventional building systems will be formulated in various domain-specific knowledge base models according to the same ontology.

(b) **Knowledge base model:** This models captures domain-specific functional knowledge based on set of instances that exemplify the conceptualization described in the ontology model. These instances are intended to support automatic classification and inference of membership necessary for querying. In this case, a small set of engineering perdurants are modeled to capture part-whole and generic causal relationships of perdurants associated with functions in two different domains, presented in the case studies developed later for validation. These relationships allow to describe generic teleological patterns, to be instantiated and incrementally refined according to most accepted domain knowledge. Hence, the intent behind the formulation of these patterns is to provide a common abstraction for the specification of functional requirements (demand side), as well as for specification of functionality of systems and components (supply side). Inference of functional interactions are expected to rely on the reuse of these patterns at various levels of refinement and expressivity.

(c) **Query/Rule model:** Regarding querying, the knowledge-base, two approaches are being explored:

i. The simplest query method is the creation of equivalent anonymous classes, which is known as DL Query. A DL query expression is based on equivalence axioms that combine logical quantification and connectives over classes and nominals. This allows OWL-DL reasoners to provide inferences both at the class level as well as at the instance level. These DL query expressions then can be stored in the ontology for future reuse.

ii. The second approach involves the use of the SPARQL Protocol and RDF Query Language (SPARQL), which is a query language for semantic databases, able to retrieve and manipulate data stored semantic graphs specified according the Resource Description Framework (RDF). SPARQL became a

standard under the World Wide Web Consortium, and it is considered as one of the key technologies of the sematic web, as showed in Figure 6.2.

The semantic integration of these three models at the front-end is a long term goal for a functional modeling framework for Building Design. To that end, a shared ontology covering the characterization of functionality from the demand, supply and evaluation sides would provide not only a more comprehensive conceptualization to enhance interoperability and automation across different domain-specific applications, but it would also provide a shared vocabulary to support more effective interdisciplinary collaboration and decision-making. In particular, the semantic integration grounded on a shared ontology of functions would contribute to make different functional perspectives more explicit to all stakeholders involved, hopefully contributing to reduce the asymmetry in functional knowledge among stakeholders in AEC industry. In the next section, an overview of the proof-of-concept implemented for the validation of the research hypothesis is presented.

6.3 Overview for proof-of-concept implementation of Aspecto-FR

The proof-of-concept implementation for the proposed functional modeling framework is named Aspecto-FR, which defines the IRI for the main ontology model and the different knowledge base models developed for the case studies. For reference, this IRI is <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#>, which is referenced by the prefix fr: in the corresponding SPARQL query expressions throughout the rest of this dissertation.

In the way this framework is intended to work in practice, the information about design elements and their structural relations are supposed to provide a starting point for the behavioral and functional characterization of the design model. Such characterization is grounded in on a series of ontological categories and relationships from DOLCE-FR, translated into OWL-DL, plus an number of additional definitions that are specific to the ontology model of this framework.

Among the main relations proposed to allow inference of cross-cutting functional participation underpinning the elucidation of aspect system are: (i) mereological relations of

Table 9: Relations (as OWL object properties)

	Entity type	DOLCE-FR	Aspecto-FR
1	Behavior	X	
2	Behavior Environment	X	
3	Mode of deployment	X	
4	CauseMD	X	
5	Device-centric function	X	
6	Environment-centric function	X	
7	Satisfiable device function		X
8	Satisfiable environment function		X
9	Realizable environment function		X
10	Aspect system		X

parthood between endurants and perdurants (i.e. *part* and *proper_part*) with role transitivity and reflexivity, (ii) specific participation between technical endurants and engineering perdurants, either qualified or not, as discussed previously (i.e. *participant_in*), and (iii) non-deterministic causal relations between perdurants (i.e. *_c_causes* and *_d_causes*), where the latter has role transitivity. Table 9 provides an overview of the most important entity types implemented in Aspecto-FR. Table 10 provides an overview of the most important object properties (i.e. relations) and the OWL-DL characteristics for each. Additional entities and relations will be introduced in the presentation of the case studies.

One of the main tasks of the implementation involved the translation of the original First-Order Logic (FOL) definitions in DOLCE-FR into OWL-DL expressions. Since DL admits only binary relationships, several of the FOL n-ary definitions required the use of modeling patterns, sometimes called OWL *reification* [329]. Many other relations were binary, allowing straightforward translation. Indeed, some were already provided by DOLCE-Lite, including *part* and *participation* relations in the form of OWL-DL object properties. However, in the proof-of-concept ontology developed here, some of these relations were modified to better support cross-cutting traversal of functional model.

These modifications involved the use of OWL property characteristics, such as transitivity, reflexivity and symmetry among others. Additionally, many relations were defined using property chains, to link properties of different types in a way not allowed by simple transitivity. Moreover, property chains are also extensively in the ontology model, in conjunction with the mechanism of *rolification* [208] to support some of the inferences required.

Table 10: Overview of most relevant object properties and property characteristics in Aspecto-FR.

		Inverse	**Functional**	**Transitive**	**Symmetric**	**Reflexive**	**Role chain**
1	part-of	X		X		X	
2	proper-part-of	X		X			
3	s_connected_to						
4	s_directly_connected_to				X		
5	s_adjacent_to				X		
6	has_component						
7	participant-in	X					
8	nominal_participant_in	X					X
9	_b_qualified_participant_in						X
10	causal_participant_in	X					X
11	_c_causes	X		X			X
12	_d_causes	X	X				
13	behavior_constraint_in_perdurant						
14	behavior_constraint_in_endurant	X					
15	_b_has_capability						X
16	functional_role_in						X

The specific definitions for the relation and entity types identified in the table are introduced in the following subsections. All the listings are presented according to the Manchester syntax [178]. Keyword declarations are in black bold black fonts (**Class**, **SubPropertyOf**), logical quantifiers are presented in mauve (some,only), along with cardinality restrictions and reference to literals (value). Logical connectives are in cyan (and, or,not), and property chains are indicated by the symbol (o). A similar color convention is used in the graphical interface of Protégé [237], the OWL ontology editor used to develop this work, and it is used throughout this document to facilitate readability. Prefixes with the source ontologies have also been omitted for the most part, except when it makes sense to illustrate the dependency on definitions from different ontologies. The complete set of ontologies developed in this research are provided in Appendices for reference.

6.3.1 Part and proper-part relations

As mentioned, the definitions of *part* and *proper-part* in OWL-DL are based on those provided by DOLCE-Lite, but with different characteristics. The listing below presents the definitions with the changes added.

Listing 6.8: OWL-DL definition of part and part-of relations.

ObjectProperty: part

SubPropertyOf:

dolce−lite:immediate−relation

Characteristics:

Transitive,

Reflexive

InverseOf:

part−of

ObjectProperty: proper−part

SubPropertyOf:

part

Domain:

dolce−lite: particular

Range:

dolce−lite : particular

InverseOf:

proper−part−of

6.3.2 Participation relations

Regarding participation, the original definition from DOLCE-Lite was adopted without any added characteristics. However, three sub-properties were added, namely nominal_participation , _b_qualified_participation and _b_constrained_participation . The first refers to a generic, unqualified participation, whereas the second refers to a functional participation, i.e. a participation nominally allocated to an artifact. Thus, while a boiler participates in hot showers, its nominal participation is in boiling the water. Hence the latter is its nominal, device-centric function.

On the other hand any form of participation, nominal, intentional or accidental, may be qualified, that is, characterized by a particular type of behavior. In this approach, adopted from DOLCE-FR, a behavior is not the phenomena being described, but a particular description of the phenomena. The term qualification also indicates that such description may belong to a type. In this way, the description can be further characterized, for instance, in terms of constraints. The latter corresponds to a constrained participation.

A different type of participation, called in Aspecto-FR a *causal participation* is described in the following subsections. Initially considered to be a specialization of the DOLCE relation $participant - in$, it was made a different property type due to implementation reasons, to be explained later. The OWL-DL definitions for these four types of participation are explained next.

- **Generic participation:** The most basic participation relation of design elements in perdurants (i.e. processes, actions or events) in the form $PC(a, e)$. The original DOLCE-Lite definitions are adopted without modification, as showed in the following listing.

Listing 6.9: OWL-DL definition of generic nominal participation.

ObjectProperty: participant−in

SubPropertyOf:

dolce−lite:immediate−relation−i

Domain:

endurant

Range:

perdurant

InverseOf:

participant

- **Nominal participation:** A participation that holds between in output perdurant and a technical artifact, in which the output perdurant has been allocated the artifact as its nominal function. The simplest, most direct form of allocation follows the same form $PC(a, e, t)$, given in 6, but without time indexation. A more complete allocation is based on the specification of device-centric functions, which may involve qualification with behavioral constraints. The axiom for nominal participation in Aspecto-FR is defined as a sub-property of $participant - in$ from DOLCE-Lite.

Listing 6.10: OWL-DL definition of nominal participation.

ObjectProperty: nominal_participant_in

SubPropertyOf:

dolce−lite−fr:participant−in

SubPropertyChain:

inverse (_DF_artifact) o _DF_outPD

Domain:

inverse (_EF_nominal_artifact) some _E_function

Range:

EPD

InverseOf:

nominal_participant

This listing introduces the first example in which property chains are used, along with use of *inverse* clauses. The symbol 'o' in the property chain clause indicates the linkage with other properties. Thus, the domain of the last linked property becomes the domain of the property itself. The inverse clauses on the other hand express that any endurant declared as artifact in the specification of a function stands in a nominal participation with the output perdurant of such function. The declaration of a nominal artifact is given in the functional specification by means of the object properties _DF_artifact and _EF_nominal_artifact. Finally, the range EPD stands for an engineering possible perdurant, defined by DOLCE-FR.

It is necessary to indicate that the validity of such participation is a matter of domain expertise, and it is not the goal of the ontology model to impose semantic constraints on what type of entities may stand in such participation. Following with the previous example, nothing prohibits the assertion that boilers have nominal participation in something else besides boiling (in hunting, for example). Domain expertise of the sort required to control these types of assertion needs to be captured in corresponding knowledge base models by means of appropriate semantic constraints.

- **Qualified participation:** Participation of design elements in perdurants where the relation is qualified by a specific type of participation. This implements the semantics of the definition $Beh(a, e, b)$ (given in 8), where b is a type of behavior, such that b stands for Behavior(b). In Aspecto-FR, this type of participation is defined as follows.

Listing 6.11: OWL-DL definition of qualified participation (qualified by behavior).

ObjectProperty: _b_qualified_participant_in

SubPropertyOf:

dolce−lite−fr:participant−in

SubPropertyChain:

_b_has_capability o behavior_constraint_on_perdurant o r_epd

This definition introduces the first example in the use of rolification, as indicated by the linked property r_epd. Rolification is a method by which a class can be treated

as if it was a property. Thus rolification allows the class R(x) to be treated as r(x,x), which means that any instance of R stands in a relation with *itself*. This is an useful method to express certain rules as axioms, instead of relying in external rule languages such as SWRL. In this case, the linked property indicates the rolification of the class of perdurants EPD. The intent is to capture the notion of an artifact that is said to possess certain (behavioral) capability. In other words, to say that an artifact has a behavioral capability b is to say two things. First, that there is a perdurant EPD in which the artifact participates, and secondly, that such participation is qualified by a behavioral constraint. The rolification for the class EPD is given by the following axiom, which exemplifies the rolification made for other entities as well.

Listing 6.12: OWL-DL axiom for Engineering Perdurant (EPD) with rolification.

Class: EPD

SubClassOf:

GEPD,

_c_causes only EPD,

proper−part only EPD,

r_epd some Self

Besides rolification, the axiom for this class also states some additional ontological constraints, such that an instance of EPD can only cause another EPD, and that all its parts are instances of EPD.

- **Constrained participation:** A sub-property of qualified participation with specification of behavioral constraints in terms of data property values. This captures participation relations where the behavioral description needs to be further characterized by some measure (i.e. quantities or locations in a quality space according to DOLCE). This allows to add more specificity in the qualified participation described. For instance, between describing the behavior of an electrical heater as simply 'consuming electricity' or being 'energy inefficient', versus or 'consuming more than 5kWh for rooms less than 30sqf.' For this purpose, the use of constrained participation needs

to rely on use of data property values, for which the use of SWRL or other rule implementations are well suited.

Listing 6.13: OWL-DL definition of constrained participation.

ObjectProperty: _b_constrained_participant_in

SubPropertyOf:

_b_qualified_participant_in

6.3.3 Causal relations

The third series of relations defined in the framework allows to describe causality in a way that is loosely related to the notion of pre and post-conditions used in SBF 4.2.1. According to DOLCE-FR, causal relations hold between perdurants only. Thus, a device is not said to cause an output (i.e. a EPD), but to participate in another perdurant which is the one that causes the output. Under DOLCE-FR, such intermediate perdurant is represented as a fusion of perdurants called a mode of deployment.

In order to support the description of multiple participation relations and functional interactions across different technical sub-systems and levels of abstraction, the notion of causal relations adopted here is by necessity very general. This is because of two reasons. First, generality is required to avoid early commitment with domain-specific theories of causality. This should be done by lower level knowledge bases, especially built to capture domain expertise.

The second reason stems from the complexity of the topic from an ontological perspective, which is beyond the scope of this research. Among the specific issues involved, there is the problem of differentiation between different causal types, and more specifically, between causality and causation, based on the types of constraints chosen for the characterization of causal relations [220].

Among these constraints, the particular type of perdurant involved is especially relevant. The specification of the DOLCE-Lite ontology provides a brief indication on this topic. According to this, each of the main four categories of perdurants in DOLCE, namely, a)

accomplishment (category `ACC`), b) *achievement* (category `ACH`), c) *state* (category `ST`), and d) *process* (category `PRO`) enables a different descriptive focus over the same phenomena. That is, each category can be considered as an 'aspect' of the perdurant being described (the process of rock erosion is given as example). In this way, a) *accomplishments* enable a causation focus; b) *achievements* enable a effectual focus; c) *states*, a condensation focus (by collapsing the time interval to an point interval); and d) *processes* provide a causality focus [140]. For implementation reasons, the interpretation adopted in this research regarding types of causal relations has been the following:

- The causation focus, enabled by reference to accomplishments (`ACC`), denotes a *contributive cause* over effects, described by achievements (`ACH`).
- The causality focus, enabled by reference to processes (`PRO`), denotes a *necessary cause* over (change) of states, each described by the category state (`ST`).

This broad nature of this interpretation have facilitated the description of causal relations across levels of abstraction, and therefore, the modeling of functional interactions and dependencies between low-level technical functions and high-level 'soft' functions. For example, it is appropriate to say that increasing relative humidity (RH) in an office space may *contribute* to thermal comfort of its inhabitants, if the increase is within certain range in relation to temperature (T). However, it is not correct to say that such increase directly *causes* thermal comfort.

Similarly, maintaining appropriate levels of thermal comfort *contributes* to an increase in productivity, but surely it does not *cause* it. Indeed, thermal comfort is not even a necessary pre-condition. The problem is that high level functions such a comfort, productivity or safety are mereological sums of perdurants, many of which are not easy to describe, explain or predict. Thus, in some cases, a high level of productivity can be achieved even without any level of thermal comfort being provided, simply because other causal relations might be involved.

Broadly speaking, the metabolic processes of plants do *necessarily cause* transpiration, and given that transpiration contributes to an increase in relative humidity, it is possible to

say that plant transpiration contributes (causally) to thermal comfort, and even productivity. The extent of such contribution however is a matter of expertise and analysis outside the scope of the definition for this relations. The important point is that causal contribution is interpreted here as a transitive relation, where necessary causes are interpreted as functional, i.e. they only admit one single value, hence no transitivity. An informal convention to indicate the lack of transitivity is to call the property as 'direct'. Hence, a necessary cause is called a *direct cause*, denoted by the name _d_causes. However, the inverse _d_caused_by is not functional, in the sense that an event may have many direct causes. For instance, the fact that both rain and irrigation directly cause the lawn to get wet is a functional relation. However, the wet lawn as directly *caused by* rain and the irrigation is not functional. The relation has more than one value. The definitions for these two causal relations are listed below, where a contributive cause is denoted by the name _c_causes.

Listing 6.14: OWL-DL definition for causal relations.

ObjectProperty: _c_causes

Characteristics:

Transitive

Domain:

EPD

Range:

EPD

InverseOf:

_c_caused_by

ObjectProperty: _d_causes

SubPropertyOf:

_c_causes

Characteristics:

Functional

Transitive

Domain:

EPD

Range:

EPD

InverseOf:

_c_caused_by

To represent the participation of indoor plants in thermal comfort, the following axiom, called causal_participant_in is defined, also based on a property chain.

Listing 6.15: OWL-DL definition for causal participation.

ObjectProperty: causal_participant_in

SubPropertyChain:

participant−in o _c_causes

InverseOf:

causal_participant

The purpose of this was to make use of property (role) chaining capabilities of OWL-DL, together with transitivity of parthood relations to support inference of indirect 'downward' participation of elements in all perdurants that their proper parts participate in. This allows to say for instance that a system participates in all the processes of its internal sub-systems.

In summary, two forms of causal relations have been implemented in the framework, for which no theoretical claims are made regarding their ontological validity. This needs to be addressed in future research.

6.3.4 Behavior environment

The notion of behavior environment, introduced by Chandrasekaran and Josephson and formalized in DOLCE-FR (presented in 20) is defined within Aspecto-FR under the general class Environment. This is turn is defined according to the following combination of equivalent and subclass axioms, in OWL-DL (Listing 6.16).

Listing 6.16: OWL-DL definition of behavior environment.

Class: Environment

EquivalentTo:

r_environment some Self

SubClassOf:

dolce−lite−fr:physical−object

This is the most general definition for the behavior environment of a function, which needs to be refined by domain-specific knowledge bases by appropriate sub-classes. For example, a template for this more specialized definition of a behavior environment has been specified in the ontology Aspecto-FR-KB, as follows:

Listing 6.17: OWL-DL knowledge base template of a behavior environment.

Class: Aspect−fr−kb: Env1

EquivalentTo:

(DOLCE−Lite−FR:weak−connection some Aspect−FR:Environment) or

(DOLCE−Lite−FR:part−of value Aspect−FR−Solar:Env1)

SubClassOf:

Environment

The use of an equivalent class axioms along with rolification allows to declare that any element asserted or inferred as part of the Environment Env1 is also a member of the class Env1. It also allows to specify that an element connected to a member of the class is also both part of the environment Env1 and a member.

Such dual condition of class membership and parthood is at the core of the reasoning mechanism implemented to support the elucidation of aspect systems. In particular, this supports the description of multiple levels of abstraction, since any part of the environment is an environment itself at a lower level, recursively. In this way, the distinction between device-centric and environment-centric functions is no longer needed, at least in theory.

Multiple environments may be instantiated within the same functional model, to support different functional viewpoints, either at different levels of granularity or from different

stages of the system life-cycle. This capability is demonstrated in the last case study (7.3).

6.3.5 Mode of deployment and associated causal models

The notion of mode of deployment $MD(e, a, a')$, formalized in DOLCE-FR in 24, is defined simply as a subclass of EPD. As such, it inherits the same constraints regarding composition and causal relations defined for the general class of engineering perdurants. In particular, not direct assertions need to be made over properties of an instance e of this class.

Instead, these are inferred whenever such instance e is referenced by the specification of an environment function. This is done (to be explained later), by an object property _EF_in_mode_of_deployment of the class _E_function, which implements the formalization of $EnvFunc(b_0, b_1, a, a', e)$ given in (26). When this happens at the instance level, the values of following property chain of the class Mode_of_deployment are inferred by the reasoner.

Listing 6.18: OWL-DL definition for output perdurant of a mode of deployment.

ObjectProperty: _MD_causal_model_OutPD

SubPropertyChain:

inverse (_EF_in_mode_of_deployment) o _EF_outPD,

Range:

EPD

In other words, the output perdurant intended by an environment-centric function is passed to the mode of deployment of that function. From here, and based on the causal relations introduced later, the entire causal chain leading to such output can be backtracked, assuming that such chain exists in the model. This capability implements the notion of causality in a model of deployment $CauseMD(e, a, b_0, b_1)$ formalized in DOLCE-FR in 25).

The initial approach was to rely on direct participation relations without qualification of b behavioral constraints $< b_0, b_1 >$. That is, only direct references to output perdurants is supported in this version, defined in OWL-DL as an object property of a mode of deployment e. This object property is called in Aspecto-FR a _MD_causal_model. Each individual perdurant related to a mode of deployment e by means of _MD_causal_model denotes a causal link of a causal model for specified output perdurants.

Listing 6.19: OWL-DL definition for causal model in mode of deployment.

ObjectProperty: _MD_causal_model

SubPropertyChain:

_MD_causal_model_OutPD o _c_caused_by

The functionality of this particular mechanism for inference of causality is demonstrated in the first and second case studies (presented in 7.1, and 7.2). Essentially, any sequence of causal relations leading to the specified output perdurant is captured by this property. In this way, entire teleological models can be built, independently of specific participation of structural entities, which can be added later. This suggests that redundancies and even impossibilities may be included in the causal chain (also called causal model).

To give a silly example, it could be stated that the causal model for a car moving is that magic is the cause for the turning of the wheels. However, the associated aspect system would not return wizards as participants, insofar such characters are not made part of the environment. The role of filtering out entities that do not exist in the environment, or that do not make sense under a disciplinary perspective, is given to the aspect system of the function. This is explained at the end of this section, along with an alternative method for the description and inference of causal models.

6.3.6 Behavior

The class of behaviors, under which behavioral descriptions of participation need to be defined, is translated simply as a direct subclass of the category quality in DOLCE-Lite. All input and output behaviors used to qualify input and output perdurants of functions belong to this general class.

Listing 6.20: OWL-DL definition of behavior.

Class: Behavior

SubClassOf:

dolce−lite−fr:quality ,

behavior_constraint_on_perdurant some EPD

Instances of this class might be associated with data property values for further characterization of behavioral constraints, such as behavioral properties of building elements defined through property sets in IFC, to be used as input values for analytical tasks. Properties of materials and components of building assemblies, such as R-values, acoustic absorption rates, friction coefficient and the like are examples of these properties. The use of property values for further characterization of behavioral constraints is demonstrated in the last case study (7.3), where a simple taxonomy of behavioral properties relevant to the design of a ballasted PV racking system for flat was developed. In particular, the case study presents an approach in which the proposed implementation of aspect systems could be used for cross-cutting traversal and controlled 'harvesting' of input values for analysis tasks.

6.3.7 Device-centric function

The formalization of a Device-centric function $DevFunc_G(a, b_0, b_1)$ in DOLCE-FR (given in 23) is translated into OWL-DL according to the listing below. The entity _P_causal_pattern is a common super class for both device and environment functions, used to describe teleological patterns with no explicit participation asserted. The device a is referred in the OWL-DL axiom by the object property _DF_artifact. However, instead of referring to the behavior constraints b_0 and b_1, a direct reference is made to the input perdurant _DF_inPD and the output perdurant _DF_outPD qualified by those behavioral constraints.

Listing 6.21: OWL-DL definition of device-centric function.

```
Class: _D_function
    SubClassOf:
        _P_causal_pattern,
        _DF_artifact only Technical_artifact ,   // Where Technical_artifact is: a
        _DF_inPD only EPD,                       // Where EPD is constrained by: b0
        _DF_outPD only EPD                       // Where EPD is constrained by: b1
```

In other words, this formulation captures the simplest, most direct reference to input and output perdurants specified by a function as intended effects, without mediation

of behavioral descriptions. The advantages and limitations of this minimal approach is demonstrated with a toy example in the first case study (7.1). However, for more complex design situations, where qualification of functional participation with behavioral constraints is required, the definition above can be enriched with the specification of a property chain axiom to obtain the output perdurant qualified by the behavior constraint b_1. The next listing shows this axiom, which relies on the rolified classes r_function and r_epd.

Listing 6.22: OWL-DL definition to obtain output perdurant from behavioral constraint.

ObjectProperty: _DF_outPD

SubPropertyChain:

b1 o r_function o r_epd

Range:

OutPD

The property b1 showed in the listing above, as well as property b0, are part of the definition of the super class _P_causal_pattern, presented below.

Listing 6.23: OWL-DL definition of super class _P_causal_pattern.

Class: aspect−fr:_P_causal_pattern

SubClassOf:

Objectified−relation,

r_function some Self

b0 some EPD,

b1 some EPD

The use of existential quantifier some in the properties b0 and b1 allows recursive nesting functions and sub-functions. Thus, b0 can be a direct reference to a proper behavior, or an indirect reference to an behavior constrained as output behavior by another device function. In this case, to obtain either an input or output behavior of an internal function, a SWRL rule is used instead of a property chain. This is done to avoid reasoning problems that arise when dealing with nested property chains. This rule, expressed in the form of a Horn clause, is the following.

$$behavior\_constraint\_on\_perdurant(?b1, ?o) \ \wedge \ \_D\_function(?f) \ \wedge$$

$$behavior\_constraint\_on\_perdurant(?b0, ?i) \ \wedge \ b0(?f, ?b0) \ \wedge$$

$$b1(?f, ?b1) \ \rightarrow \ \_DF\_outPD(?f, ?o) \ \wedge \ \_DF\_inPD(?f, ?i)$$

6.3.8 Environment-centric function

The formalization of an environment-centric function $EnvFunc(b_0, b_1, a, a', e)$ in DOLCE-FR (given in 26) is translated into OWL-DL according to the listing below, where $EncFunc$ is called _E_function, and the device a is referred by the object property _EF_nominal_artifact. Recall that an aspect system is a subset of the behavior environment, and that the nominal artifact is the entity usually allocated such function. For instance, an HVAC is usually allocated the function of providing thermal comfort, but clearly this can only be achieved by integration of various elements of the behavior environment. If considered within a spectrum, the aspect system for thermal comfort includes, at the minimum, the HVAC itself. At the maximum, the entirety of the behavior environment is the aspect system of the function. Therefore a flexible representational approach is needed to selectively pick parts of the environment according to different performance criteria.

Listing 6.24: OWL-DL definition of environment-centric function.

Class: _E_function

SubClassOf:

_P_causal_pattern,

_EF_inPD only EPD, // Where EPD is constrained by: b0

_EF_outPD only OutPD, // Where EPD is constrained by: b1

_EF_nominal_artifact only Technical_artifact , // Where Technical_artifact is : a

_EF_in_environment only Environment, // Where Environment is: a'

_EF_in_mode_of_deployment only Mode_of_deployment, // Where _mode_ is: e

inverse (_AS_aspect_of_function) only Aspect_system

These auxiliary features allowing recursive functional calls between device-centric function are also included in the implementation of environment-centric functions, albeit with

minor modifications. In particular, an environment function can invoke the output perdurant constrained by the behavior b1 of either a device function, or another environment-centric function. These capabilities are demonstrated in the second and third case studies (7.2 and 7.3). To avoid repetition, they will not be discussed here.

In this translation however, the specification of intentionality of an agent is omitted for simplicity of the model. For now, it is assumed that the function has an association with conventional group of agents or stakeholders. The main issue being addressed is how to capture the web of interactions that may lead to potential conflicts, and not the assessment of the conflicts themselves, as this is an external evaluation task. In this regard, notice the inverse relation to the class of aspect systems. This will be explained in the following, final subsection.

6.3.9 Aspect system

During the development of this proof of concept, two approaches for the representation of aspect system have been implemented. The first relies on the OWL-DL definition for mode of deployment and casual model presented above. Specifically, the inference of participants is based on the causal links obtained by the property chain _MD_causal_model. The definition for this first alternative is as follows:

Listing 6.25: OWL-DL definition of aspect system based on property chain (alternative 1).

Class: Aspect_system

SubClassOf:

Environment,

_AS_in_mode_of_deployment only EPD, // Where EPD is: e

_AS_nominal_artifact only Technical_artifact , // Where EPD is: a

_AS_environment only Environment, // Where EPD is: a'

_AS_aspect_of_function only _E_function, // Auxiliary relation to function

proper−part−of only Environment // Consequent

The auxiliary property _AS_aspect_of_function requires the target environment function

to be asserted at the instance level, whereas the rest of the properties are inferred automatically. The assertion of the target function allows an instance of the aspect system to obtain the intended effect of the function (i.e. its output perdurant), as well as the mode of deployment associated with it. By instantiating this relationship, the following property chain with rolified classes enables the reasoner to obtain all co-participants in the function that are part of the behavior environment.

Listing 6.26: OWL-DL property chain to obtain co-participants in function.

ObjectProperty: _AS_co−participant

SubPropertyChain:

_AS_in_mode_of_deployment o _MD_causal_model o participant o r_environment

Range:

Environment

This effectiveness of this approach has demonstrated in the first case study (7.1), where the main limitations have been also been identified. In particular, this approach is not suitable where qualification of behavioral constraints is required, as it operates directly over input and output perdurants. Some workarounds have been attempted but not without impacting severely the performance of the DL reasoner. Furthermore, the inference of causal links by means of property chains has returned incomplete results, whenever nesting of functions are involved. For this reason a second approach has been implemented, based on the an equivalent class axioms, which takes the following general form.

Listing 6.27: OWL-DL definition of aspect system as equivalent class (alternative 2).

Class: Aspect_system

EquivalentTo:

(part some (participant−in some

(_c_causes some (inverse (_ED_outPD) some _E_function))))

and (proper−part−of some Environment)

SubClassOf:

proper−part−of some Environment

In this method, the causal model associated with the mode of deployment, which was specified in the first method by a property chain, here is specified internally, according to the clause (_c_causes some (inverse (_ED_outPD). The condition that participants have to be exist in the behavior environment of the function is specified in the final clause. In this way, theoretical or imaginary participants are excluded from the aspect system. Moreover, not only behavioral constraints can be associated directly, through the direct reference to the function, but recursion of nested functions works better in terms of reasoning performance and consistency of results in all preliminary tests made so far.

A demonstration is provided in the last case study, particularly in the design scenario 2 (7.3.4), where this second method is used to formulate the aspect systems of seven different environment-centric functions with nested functions, in two different behavior environments corresponding to different life-cycle stages of a PV system, namely installation and operation. The listing below exemplifies the specification of an aspect system according to the second method.

Listing 6.28: OWL-DL definition of aspect system as defined class (alternative 2).

Class: Aspect_system

EquivalentTo:

(part some (participant−in some

(_c_causes some (inverse (_ED_outPD) value _fe2.1_squaring_function))))

and (proper−part−of some Environment)

SubClassOf:

proper−part−of value _as2.1

Informally, the axiom means that if an entity has participation in some contributive cause for the output perdurant of the squaring function, and if such entity is a proper-part of the environment, then the entity is a member of this particular aspect system, defined as an equivalent class. The subclass axiom in the last clause of this definition states that if an individual is a member of the class, then it is also a proper part of the individual _as2.1, which is the abstraction aggregating the parts of the aspect system.

This last clause was an attempt to make a direct translation of the first-order logic

definition of aspect systems given in 25. For some reason however, the proper−part−of clause has a severe impact in the performance of the reasoner, increasing the running time significantly. Because of this, the subclass clause has been removed from the axiom above.

While the issue may be related with transitivity of proper-part properties, the exact problem remains unclear. In any case, it became evident that such clause was unnecessary at this level of implementation for two main reasons. First, the equivalent class definition for an aspect system really works as a query. Indeed, while in the case studies 2 and 3 these classes are kept permanently in the ontology, they can also be used in the form of DL-queries, which do not keep them in memory, thus improving reasoning performance significantly. This makes the aggregation of participants into a proxy abstraction using the same logic somewhat inefficient, when in practice this could be better handled procedurally by an external client application.

Finally, the last relation defined in Aspecto-FR allows the inference of functional interactions across multiple levels of abstraction when no complete causal model is known or can be provided. This is especially important to describe dependencies between low-level functions, and high level functions. This is because often high level functions, sometimes called 'soft' functions may refer to output perdurants with a complex internal composition, which cannot be easily broken down, or conventional causal relations to be established in a straightforward manner.

For that purpose, composition relations between perdurants can be applied in conjunction with contributive causes, so to bridge the gap between levels where causal models are available, with those that are not. In this way the meaning of *function as a role* can be implemented operationally as an object property in OWL-DL, exemplifying the intuition that this is the most general meaning for the concept of function. The following listing provides the property axiom, based on a property chain with rolification of target output perdurant r_outpd.

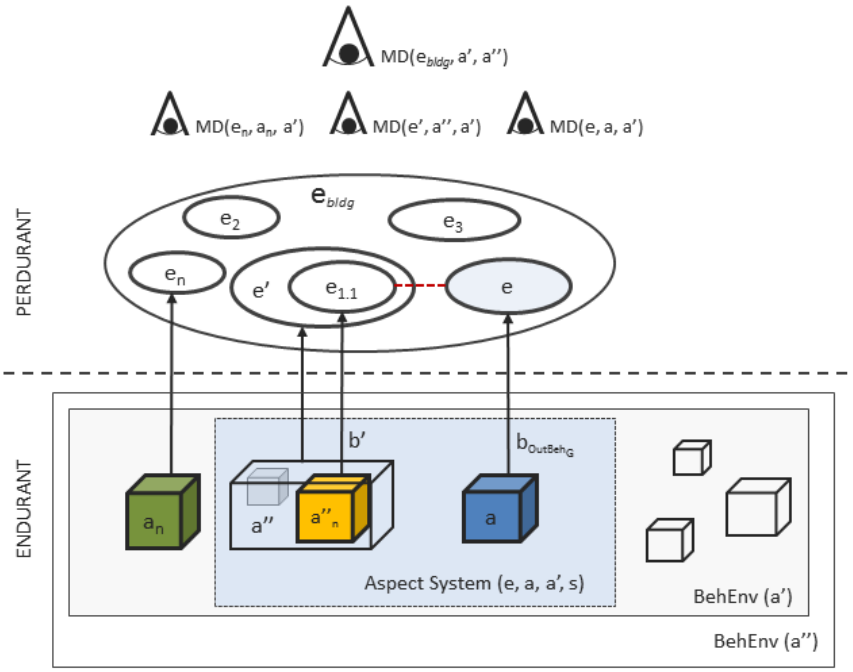

Figure 6.3: Aspect system of cross-cutting elements. Co-participation through propagation of functional roles inferred by means of transitivity of composition and casual relations.

Listing 6.29: Functional role relation, based on causal contribution and composition.

```
1 ObjectProperty: functional_role_in
2     SubPropertyChain:
3         _c_causes o proper−part−of o r_outpd,
4         proper−part−of o r_outpd
```

Now it is possible to reformulate the definition of aspect system given above, so that the relationship between low and high level functional viewpoints can be formalized, allowing the inference of vertical interactions at instance level. An example of this is provided in case study 4, in the concluding chapter (in 8.2)

Listing 6.30: Aspect system for vertical integration based on functional roles.

```
1 (part some (participant−in some (_c_causes some
2                 ( functional_role_in  some
3                     (inverse  (_EF_outPD) some _E_function)))))
4 and (proper−part−of some Environment)
```

CHAPTER VII

FRAMEWORK VALIDATION: CASE STUDIES

In this chapter, three case studies are presented, which were developed for testing of the proof-of-concept implementation, and validation of the research hypothesis. Each case is introduced in the context of a specific design problem, with a series of design alternatives and decision making scenarios that illustrate the specific requirements for a functional modeling framework with inference capabilities, as proposed by this research. The scenarios also allowed an iterative cycle of testing and refinement of the proof-of-concept implementation, which eventually led to a significant improvement of the underlying ontology model as well.

Given that the goal is to provide computational support in the identification of functional interactions and inter-dependencies that are fundamentally cross-cutting, both in terms of levels of abstraction, and in terms of life-cycle requirements, the case studies are developed taking in consideration different functional viewpoints. The possible down-side is that many of these viewpoints may not be described with the level of proficiency of a domain expert, and therefore some errors and omissions are possible. Therefore, an effort has been made to make explicit basic modeling assumptions, constraints and boundary conditions.

After introducing the design problem, and the main scenarios and design tasks involved, the approach adopted in the formalization of the model is presented. This process includes the description of the main structural and functional aspects of the design domain. For now, only an abstract, logic representation of the structure of the design is provided, with no integration with geometric data from CAD or BIM models.

The process of formalization of the model is based on a series of assertions about facts that are known regarding the structural composition and behavior of relevant parts. These assertions are made in OWL-DL using Protégé, an open-source ontology editor [237], and presented according to the Manchester syntax [178]. Additionally, class definitions and SWRL rules, discussed in the previous chapter, will be presented when necessary. Most

relevant queries, including several that demonstrate the criteria formulated in the hypothesis, are introduced. These are made using either DL query expressions, based on equivalent class axioms, or using SPARQL as query language.

The first case study is based on a simplification of a design problem concerned with the inter-dependencies between structural, thermal and manufacturing requirements of an electronic device. This example was first introduced in Chapter I, to illustrate some of the theoretical and practical dimensions of the main research problem addressed.

The second and third case studies are based on the design of a ballasted PV racking system for commercial flat-roofs. This example was also introduced in the first chapter, and it was chosen as case study because it exemplifies a realistic set of design requirements and conditions associated with functional integration and multi-functionality of components that constitute the core of the research hypothesis. Such conditions reflect an effort to go beyond simple 'toy problems', by testing the resources implemented in the framework to describe interactions across different life-cycle performance requirements.

Moreover, in order to demonstrate certain level of generality regarding the coverage of different use case scenarios and design problems, a fourth case study has been developed. This will be used to as part of the final overview and discussion offered in the concluding chapter, providing a series of concrete references from the implementation side about the main theoretical aspects addressed by the research. To do so, this fourth case study elaborates on a series of small examples given in previous chapters, related to open office settings, and the interactions that emerge between different building subsystems, with impact in Indoor Air Quality (IAQ) and productivity.

Obviously, all the functional models presented are still simplifications, and more work has to be done to evaluate and refine both the theoretical and implementation aspects of the proposed framework. Nevertheless, these models along with the proof-of-concept implementation provide a valuable platform for experimentation and research, allowing the identification of a series of viable approaches to tackle more realistic and complex functional modeling scenarios.

At the end of this chapter, a general summary for all case studies will be provided, along

with main findings, strengths and limitations. Main metrics associated with each model are included, along with performance in tasks involving reasoning and query. Recommendations and directions will be given at the end regarding future work.

7.1 Case study 1: Electronic device

The first case study is based on the simple example of an electronic device provided in Chapter I (1.1). In the example, the problem was framed in terms of functional conflicts arising by increasing the thickness of the device casing to improve its resistance to impact. By doing so in the CAD environment, the designer might not be aware of the functional implications regarding other requirements, such as thermal dissipation and manufacturability of the casing. While the CAD model can have parametric constraints that may enforce dimensional limits to the geometry of the design, the functional semantics or rationale behind these limits are often opaque.

This example was modeled following the formalism proposed in the specification of the Aspecto-FR framework. The main goal of this first model is to study the applicability of a minimum set of logical axioms and rules. Figure 7.1 provides a schematic illustration of the device, labelled according to the convention used in the implementation of the functional model in OWL-DL.

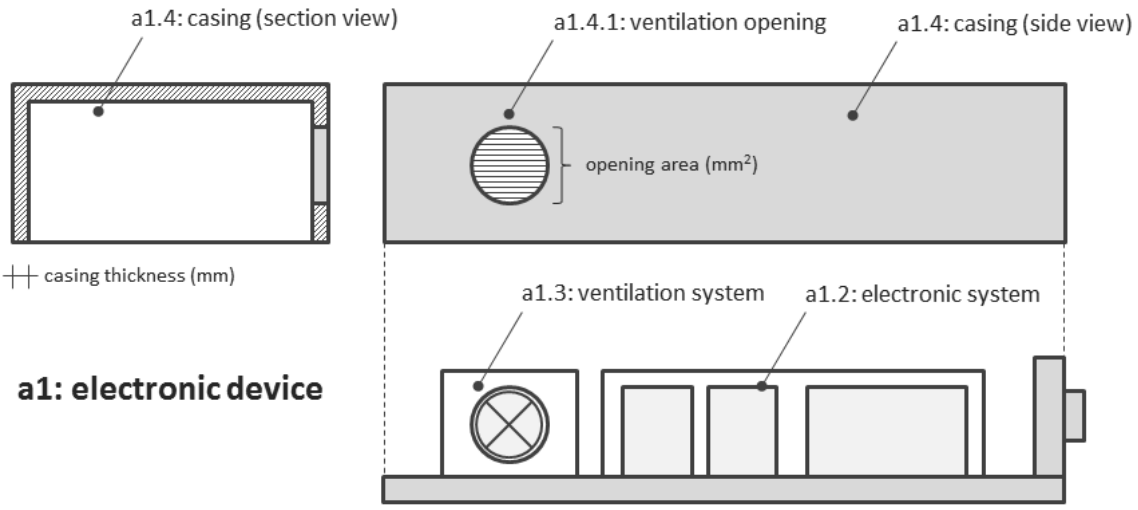

Figure 7.1: Schematic drawing of electronic device with part labels.

The problem is framed according to five main functional perspectives or requirements, each at a different level of granularity regarding the overall functionality of the device. These requirements have been simplified for the sake of the example, and roughly allocated to different sub-systems and parts of the device, according to their main nominal function. These functional viewpoints are:.

1. The overall device function (e.g. an amplifier, a computer, etc.)
2. Electronic sub-function (e.g. the function of the circuit board)
3. Ventilation sub-function (e.g. heat sink and ventilation fan)
4. Structural sub-function (e.g. impact resistance of the plastic casing)
5. Manufacturability (e.g. injection molding requirements)

It is important to note that manufacturing processes are considered within the theoretical framework of this research as a type of design function, insofar they can be associated to specific forms of output perdurants. Thus, a plastic casing or other sort of artifact has to be designed to allow for certain manufacturing processes to occur, in predetermined ways. The same observation applies to other logistic requirements, such as packing, storage or transportation, among other life-cycle requirements for which a measure of performance can be defined.

For simplification purposes, the explicit reference to functions, as defined by DOLCE-FR, has been avoided in this case study. In turn, the model makes reference directly to output perdurants (e.g. $e1\_outPD$), which describe the intended functional effects of the different components of the device without further qualification of behavioral constraints. In this way, the proposed use of relations of parthood, participation and causation become more transparent for the purposes of this first example.

7.1.1 Requirements and design scenarios

The main task of the functional model implemented is to capture interaction patterns associated with two types of design modifications. The first modification is an increase in

the thickness of the device casing to improve impact resistance. This change may lead to additional levels of heat containment that could affect negatively the functionality of the electronic components. To mitigate such unintended side-effect, the area of the ventilation opening might be enlarged to improve air flow, but this second modification impacts the overall strength of the casing. Furthermore, a thicker casing may violate structural constraints associated with manufacturing requirements, such as dimensional tolerances of a given injection molding process. A final option is to reduce the heat output of the electronic components, such that the ventilation opening might be smaller, but this can reduce air flow requirements, leading to internal overheating.

The following table shows the initial design parameters and the modifications discussed. Each set of parameters conforms a different design scenario for which reasoning over functional relationships was performed in the model.

Table 11: Parameter values for four different design scenarios.

Parameter	**Scenario 1**	**Scenario 2**	**Scenario 3**	**Scenario 4**
Casing thickness:	12 mm	15.5 mm	12 mm	12 mm
Vent opening area:	26.25 mm^2	26.25 mm^2	35 mm^2	17.5 mm^2
Heat output:	15 Watts	15 Watts	20 Watts	10 Watts
Heat/opening factor:	1.75x	1.75x	1.75x	1.75x

7.1.2 Problem formalization and encoding

The functional model developed for this case study is very simple, involving a small knowledge base with only four SWRL rules. No additional class axioms or object property axioms were specified beyond those formulated in the Aspecto-FR ontology. The intent was to use the most minimalistic version of the framework, which relies only on basic participation relations without the use of behavioral constraints or more complex specifications of functionality. Therefore, most of the assertions and inferences occur at the instance level, focusing on the transitivity of parthood and causal relations initially asserted between perdurants, and the participation of components on those perdurants. The following subsection describes these initial assertions, followed by a description of the SWRL rules adopted for reasoning of functional interactions. The complete set of definitions is provided in Appendix D.

7.1.2.1 Individual axioms

Individual axioms are assertions about facts at the instance level. In this model, facts were asserted regarding two main types of entities, namely, endurants and perdurants. Assertions about endurants include mostly structural relationships of composition, i.e. part-of relations. These were represented logically in the model, but are expected to be provided in the future entirely by a CAD model, along with other relevant geometric information.

Other assertions involve participation of endurants in perdurants, as well as composition and causality between perdurants. Figure 7.1.2.1 shows a matrix with asserted functional allocations and parthood. Rows indicate parts and components of the device, while columns indicate perdurants. As mentioned before, functions are allocated to different parts of the device by means of participation relations only. In particular, intended functional participation in output perdurants is called 'nominal' participation. This t is denoted by dark cells with a white `N` letter. Other forms of participation leading to known contributions and side-effects, are denoted by a `P` letter.

Assertion of composition relations between components of the device a is denoted by the indices $a1$, $a1.4$, $a1.4.1$, etc., and according to the schematic drawing showed above (7.1). A similar convention is used to describe the assertion of composition relations between perdurants e, denoted by the indices $e1$, $e1.4$, $e1.4.1$, etc. The main difference however is that third level sub-indices indicate a causal link leading to the correspondent output perdurant. Thus, $e1.3\_outPD$ indicates an intended output perdurant, i.e. a function, the is obtained from the preceding causal chain starting nominally at $e1.3.1$. While this data structure is similar to a linked list proposed for OWL [96], it is not forced to be delimited at the initial and terminal nodes. This allows causal relations to be made with external nodes at 'midspan' of the chain, in order to capture side-effects and interactions coming in or out of the local system.

7.1.2.2 Rules

The knowledge base capturing the design domain consists of four rules described below. The clauses omit the prefix of the source ontologies providing class and property definitions, for

		device interface (inputs)	device function	electronic sub-processes	electronic sub-function	radiant heat	convective heat	heat containment	air flow / heat release	ventilation sub-function	loading / impact	load distribution	impact absorption	structural sub-function	casing manufacturability
		e1		e1.2		e1.3					e1.4				e2
initial		e1.1	e1_outPD	e1.2.1	e1.2_outPD	e1.3.1	e1.3.2	e1.3.3	e1.3.4	e1.3_outPD	e1.4.1	e1.4.2	e1.4.3	e1.4_outPD	e2_outPD
electronic device	a1		N												
electronic subsystem	a1.2				N										
ventilation subsystem	a1.3									N					
casing	a1.4													N	P
ventilation opening	a1.4.1								N						
air									P						

Figure 7.2: Matrix of individual assertions regarding composition, functional participation and causality. Rows indicate parts of the device, while columns indicate perdurants.

illustration purposes. These are provided by the ontologies DOLCE-Lite-FR, Aspecto-FR or Aspecto-FR-KB described earlier. The only exception is the inclusion of the *swrlb* prefix referring to SWRL built-in operators for parametric evaluation of the rules. [1].

The first rule specifies the area of the ventilation opening $a1.4.1$, part of the casing $a1.4$. In this rule, the ventilation opening is referred to by its class, denoted by the clause $Ventilation\_opening(?v)$. The rule specifies the area of the ventilation opening as a function of the heat output produced by the electronic subsystem. The heat output is specified by the clause $q\_heat\_output(?s, ?h)$, where $?s$ is the electronic subsystem, and $?h$ is the heat output, in Watts (the units are omitted). The area is calculated by the clause $swrlb : multiply(?a, ?h, 1.75)$, which multiplies the heat output h by a factor of 1.75x.

[1] From http://www.w3.org/2003/11/swrlb#

$$Ventilation\_opening(?v) \wedge Electronic\_subystem(?s) \wedge q\_heat\_output(?s, ?h) \wedge$$
$$swrlb : multiply(?r, ?h, 1.75) \rightarrow has\_area(?v, ?r)$$

The second rule triggers the participation of the ventilation opening $a1.4.1$ in the functionality of the electronic subsystem $e2\_outPD$ if its area is less than 20 (i.e. 20 mm^2). Notice that the output perdurant is referred to by its name, that is, as a constant, and not by its class. Additionally, because the ventilation area is so small, the causal participation of air is also established (e.g. air temperature along other properties).

$$Ventilation\_opening(?v) \wedge has\_area(?v, ?r) \wedge swrlb : lessThan(?r, 20) \rightarrow$$
$$causal\_participant\_in(?v, e1.2\_outPD) \wedge causal\_participant(air, e1.2.1)$$

The third rule triggers the participation of the ventilation opening $a1.4.1$ in the loading distribution of the casing, which affects its impact resistance and overall structural performance. This occurs if the opening area is larger than 30 mm^2, which also leads to the causal participation of the electronic subsystem in the impact resistance to be established.

$$Ventilation\_opening(?v) \wedge has\_area(?v, ?r) \wedge swrlb : greaterThan(?r, 30) \rightarrow$$
$$causal\_participant\_in(?v, e1.4.1) \wedge causal\_participant\_in(a1.2, e1.4.2)$$

The fourth, and final rule involves evaluation of the casing thickness parameter. The casing is referred in the rule not by its class, but directly as a nominal using the symbol $a1.4$. The thickness parameter is denoted by the symbol $?x$. If the thickness exceeds 15 mm, then the casing $a1.4$ becomes a causal participant in heat containment process, denoted by the perdurant nominal $e3.3$. In other words, the rule triggers the participation of the casing

in heat containment, which is part of the overall process of heat transfer taking place as part of the overall functionality of the device.

$$has\_thickness(a1.4, ?x) \wedge swrlb : greaterThan(?x, 15) \rightarrow$$

$$causal\_participant\_in(a1.4, e1.3.3)$$

It is important to note that the participation relations triggered by the rules occur most of the time under normal circumstances. That is, casings normally participate in heat containment, and openings normally affects the load bearing capabilities of the casing. However, while this may seem trivial, the rules allow to formalize the specific conditions under which these relationships need to be made explicit in the model.

7.1.3 Design trade-off scenario 1

The first scenario is based on the initial design values, and the assertions made in the model. For convenience, the table is presented again without the values of the following tests.

Table 12: Design parameters scenario 1: initial values

Parameter	**Scenario 1**	**Scenario 2**	**Scenario 3**	**Scenario 4**
Casing thickness:	12 mm	-	-	-
Vent opening area:	26.25 mm^2	-	-	-
Heat output:	15 Watts	-	-	-
Heat/opening factor:	1.75x	-	-	-

The reasoner Pellet was used within Protégé to generate inferences based on the initial assertions and parametric values. In particular, this allows to make explicit the causal participation of the device parts in various sub-processes or states. The matrix below shows the inferences produced. Inferred causal participation is denoted by the letter `C`, whereas `P` denotes the inference of a component participating in the internal processes of a part. This condition, originally defined in the Aspecto-FR-KB ontology, is based on the transitivity of parthood relationships, and establishes that an artifact always participate in the perdurants in which its parts participate. This can be seen in the row $a1$, where the participation of the entire device in the perdurants of its internal parts is inferred.

		device interface (inputs)	device function	electronic sub-processes	electronic sub-function	radiant heat	convective heat	heat containment	air flow / heat release	ventilation sub-function	loading / impact	load distribution	impact absorption	structural sub-function	casing manufacturability
		e1		**e1.2**		**e1.3**					**e1.4**				**e2**
test 1		e1.1	e1_outPD	e1.2.1	e1.2_outPD	e1.3.1	e1.3.2	e1.3.3	e1.3.4	e1.3_outPD	e1.4.1	e1.4.2	e1.4.3	e1.4_outPD	e2_outPD
electronic device	a1	P	N	P	P	P	P	P	P	P	P	P	P	P	P
electronic subsystem	a1.2	P	C	C	N	C	C	C							
ventilation subsystem	a1.3									N					
casing	a1.4								P	C	P	P	P	N	C
ventilation opening	a1.4.1								N	C					
air									P	C					

Figure 7.3: Matrix of inferred relationships under initial parametric values.

The inference of participation in the perdurants of internal parts also can be seen in the row $a1.4$, which indicates that the casing also participates in the air flow perdurant $e1.3.4$, asserted originally as the nominal function of the ventilation opening $a1.4.1$. Notice that, as discussed in the hypothesis, the columns represent co-participation of endurants in the same perdurant. Therefore, whenever a perdurant is intended by design as an output perdurant, that is, a simplified description of a required function, the corresponding column denotes the Aspect System of the function. Conversely, multiple participation of the same endurant in different output perdurants indicate the possibility of functional interactions across different subsystems.

7.1.4 Design trade-off scenario 2

The second scenario elaborates in the increase of the casing thickness from 12 mm to 15.5 mm, while keeping the other parameters the same. Table 13 provides an overview of the new parameter values used in this scenario.

The matrix in figure 7.1.4 shows the results after running the reasoner. Inferences about

Table 13: Design parameters scenario 2: increased thickness

Parameter	Scenario 1	Scenario 2	Scenario 3	Scenario 4
Casing thickness:	-	15.5 mm	-	-
Vent opening area:	-	26.25 mm^2	-	-
Heat output:	-	15 Watts	-	-
Heat/opening factor:	-	1.75x	-	-

the causal participation of the casing with increased thickness is showed with the added `C` letters in the row corresponding to the casing part ($a1.4$). This inference was triggered by the fourth rule in the knowledge base (rule 7.1.2.2), which enforces the participation of the casing in the process of heat containment. Again, the assumption made is that such participation is constant whenever the device is working. However, the purpose of the rule is to make the participation explicit, long with the possible functional implications, as soon as certain threshold values have been crossed. The evaluation of the actual impact in performance is a different task, which can be informed by the activation of the rule. In particular, the goal is to identify and collect the design elements and their properties that need to be used as input set for an analysis application or simulation tool.

		device interface (inputs)	device function	electronic sub-processes	electronic sub-function	radiant heat	convective heat	heat containment	air flow / heat release	ventilation sub-function	loading / impact	load distribution	impact absortion	structural sub-function	casing manufacturability
		e1		**e1.2**		**e1.3**					**e1.4**				**e2**
Scenario 2		e1.1	e1_outPD	e1.2.1	e1.2_outPD	e1.3.1	e1.3.2	e1.3.3	e1.3.4	e1.3_outPD	e1.4.1	e1.4.2	e1.4.3	e1.4_outPD	e2_outPD
electronic device	a1		N	C	C	C	C	C	C	C	P	P	P	P	P
electronic subsystem	a1.2	P	C	C	N	C	C	C		C					
ventilation subsystem	a1.3					P	P	P	P	N					
casing	a1.4		C	C	C	C	C	C	P	C	P	P	P	N	C
ventilation opening	a1.4.1								N	C					
air									P	C					

Figure 7.4: Matrix of inferred relationships with new thickness value of 15.5 mm.

Table 14: Design parameters scenario 3: increased heat output

Parameter	Scenario 1	Scenario 2	Scenario 3	Scenario 4
Casing thickness:	-	-	12 mm	-
Vent opening area:	-	-	35 mm^2	-
Heat output:	-	-	20 Watts	-
Heat/opening factor:	-	-	1.75x	-

7.1.5 Design trade-off scenario 3

The third design scenario involves a change in the heat output of the electronic subsystem. Because the area of the ventilation opening in the casing is dependent on the heat output value, either rule 7.1.2.2 or rule 7.1.2.2 might get activated. In the first case, the opening area is reduced due to a small heat output. This modification may impact the performance of the electronic components (denoted by $e1.2\_outPD$) due to a lack of proper ventilation.

In the second case, the area is increased to improve air flow and ventilation to compensate for a higher heat output. However this may lead to a reduction in the impact resistance of the casing and its general strength (denoted by $e1.4.1$). Table 14 shows the parameter values for this scenario. In this case, the increase in the heat output, from 15 to 20 Watts leads to opening area of 35 mm^2, whereas before it was 26.25 mm^2 (20 Watts x 1.75). The thickness was set back to the original 12 mm.

The results of the reasoner are presented in matrix below. Notice that the columns of each output perdurant actually represent the Aspect System for that functional requirement. By comparing the sequence of matrices from scenario 1 to scenario 3, the incremental elucidation of Aspect Systems motivating the hypothesis of this research can be visualized.

As the design changes or evolves, cross-cutting subsets of components are aggregated under a given functional perspective. For instance, the Aspect System for output perdurant $e1.2\_outPD$ has changed from scenario 1 to scenario 3, first with the addition of the casing, and then with its removal.

An observation is needed regarding the composition hierarchy of the device. It may be argued that the ventilation opening, while being a structural part of the casing, belongs functionally to the ventilation subsystem (alongside with heat sinks and ventilation fans, not included in the example). The observation suggests a lattice composition structure,

Scenario 3		e1.1	e1_outPD	e1.2.1	e1.2_outPD	e1.3.1	e1.3.2	e1.3.3	e1.3.4	e1.3_outPD	e1.4.1	e1.4.2	e1.4.3	e1.4_outPD	e2_outPD
		device interface (inputs)	device function	electronic sub-processes	electronic sub-function	radiant heat	convective heat	heat containment	air flow / heat release	ventilation sub-function	loading / impact	load distribution	impact absortion	structural sub-function	casing manufacturability
		e1		e1.2		e1.3					e1.4				e2
electronic device	a1		N	C	C	C	C	C	C	C	C	C	C	C	C
electronic subsystem	a1.2	P	C	C	N	C	C	C		C					
ventilation subsystem	a1.3					P	P	P	P	N					
casing	a1.4								P	C		C	C	N	C
ventilation opening	a1.4.1								N	C		C	C	C	
air									P	C					

Figure 7.5: Matrix of inferred relationships with new heat output value of 20 Watts, and increased ventilation area of 35 mm^2 .

instead of a single tree hierarchy as specified in the original model.

By making the ventilation opening now part of both the ventilation subsystem, as well as the casing, a new set of causal participation relations are inferred. These new relations are showed in the Figure 7.1.5 with red `C` letters.

The added participation relations lead to a new Aspect System for the structural function of the casing. Indeed, this is consistent with the intuition that larger openings required to improve air flow may affect the structural performance of the casing. Now, there might be a need to increase the thickness of the casing back to 15 mm, to compensate for the larger opening. The functional implications of this change, similar to those of scenario 2, are showed in the same matrix with the red `C` letters next to the casing $a1.4$. With this design change, the participation of the casing spans across multiple functions. In other words, the casing becomes the most functionally integrated component of the device.

Scenario 3b		device interface (inputs)	device function	electronic sub-processes	electronic sub-function	radiant heat	convective heat	heat containment	air flow / heat release	ventilation sub-function	loading / impact	load distribution	impact absorption	structural sub-function	casing manufacturability
		e1		e1.2		e1.3					e1.4				e2
		e1.1	e1_outPD	e1.2.1	e1.2_outPD	e1.3.1	e1.3.2	e1.3.3	e1.3.4	e1.3_outPD	e1.4.1	e1.4.2	e1.4.3	e1.4_outPD	e2_outPD
electronic device	a1	C	N	C	C	C	C	C	C	C	P	C	C	C	C
electronic subsystem	a1.2	C	C	C	N	C	C	C		C		C	C	C	
ventilation subsystem	a1.3					P	P	P	P	N	C	C	C	C	
casing	a1.4		C	C	C	C	C	C	P	C	P	C	C	N	C
ventilation opening*	a1.4.1								N	C		C	C	C	
air									P	C					

Figure 7.6: Matrix of inferred relationships for new composition hierarchy, and increased casing thickness.

7.1.6 Design trade-off scenario 4

The last scenario involves reducing the heat output of the electronic subsystem, perhaps by making it smaller or more energy efficient. By reducing the heat output specification to 10 Watts, the area of the ventilation opening in the casing is reduced to 17.5 mm^2. While this allows to reduce the thickness of the casing back to the original 12 mm, it may cause its own problems related with the possibility of insufficient air flow for cooling, for instance, due to dust accumulation and clogging. Rule 7.1.2.2 triggers the causal participation of the reduced ventilation opening in the functionality of the electronic subsystem. The causal chain related to dust accumulation and clogging could be added in the future for explanatory purposes.

Table 15 presents the current values used in this last scenario, and the results of the reasoning process are presented in the following matrix (7.1.6). In red, the causal participation inferred by making the ventilation opening part of both the casing and the ventilation

Table 15: Design parameters scenario 4: decreased heat output and opening area

Parameter	Scenario 1	Scenario 2	Scenario 3	Scenario 4
Casing thickness:	-	-	-	12 mm
Vent opening area:	-	-	-	17.5 mm^2
Heat output:	-	-	-	10 Watts
Heat/opening factor:	-	-	-	1.75x

Scenario 4		device interface (inputs)	device function	electronic sub-processes	electronic sub-function	radiant heat	convective heat	heat containment	air flow / heat release	ventilation sub-function	loading / impact	load distribution	impact absorption	structural sub-function	casing manufacturability
		e1		e1.2		e1.3					e1.4				e2
		e1.1	e1_outPD	e1.2.1	e1.2_outPD	e1.3.1	e1.3.2	e1.3.3	e1.3.4	e1.3_outPD	e1.4.1	e1.4.2	e1.4.3	e1.4_outPD	e2_outPD
electronic device	a1	C	N	C	C	C	C	C	C	C	P	C	C		C
electronic subsystem	a1.2	C	C	C	N	C	C	C		C		C	C		
ventilation subsystem	a1.3				C	P	P	P	P	N	C	C	C		
casing	a1.4		C	C	C	C	C	C	P	C	P	C	C	N	C
ventilation opening*	a1.4.1		C	C	C	C	C	C	N	C					
air					C				P	C					

Figure 7.7: Inferred matrix for scenario 4. Heat output of 10W and area of 17.5mm^2.

subsystem. As can be seen, the Aspect System for the electronic sub-function became more complex. For a small ventilation opening, environmental conditions, and in particular, air contaminants, particles, air humidity and temperature may need to be considered with more attention during analysis of performance.

7.1.7 Queries and evaluation

To summarize the results of the different scenarios tested, an overview is presented regarding the specific queries performed over the model using SPARQL. The first listing presents a query about the perdurants (e.g. events or processes) leading to the intended output perdurant $e1.2\_outPD$.

Table 16: Query results for causal chain for sub-function $e1.2\_outPD$.

dev:e1.2_outPD	fr:_d_causes	dev:e1_outPD
dev:e1.1	fr:_d_causes	dev:e1.2
dev:e1_inPD	fr:_d_causes	dev:e1.1
dev:e1.2.1	fr:_d_causes	dev:e1.2_outPD

Listing 7.1: SPARQL query 1. Causal chain for $e1.2\_outPD$.

```
PREFIX fr:<http://www.ou.edu/coa/ontologies/2018/6/Aspecto-FR#> PREFIX dc:
<http://www.ou.edu/coa/ontologies/DOLCE-Lite-FR.owl#> PREFIX dev:
<http://www.ou.edu/coa/ontologies/2018/4/Aspecto-FR-KB-Device#> SELECT ?x ?y ?z
WHERE
{  ?z rdf:type fr: EPD . # EPD is an engineering perdurant. From DOLCE-FR.
   ?y rdf:type owl:ObjectProperty .
   dev:e1.2_outPD fr:_c_caused_by ?x .
      ?x fr:_d_causes ?z ;
   ?y ?z .
   FILTER(?y = fr:_d_causes) }
```

The results of the query are showed in the table 16 without any specific causal order. The first column on the left shows the perdurants in generic causal relation with the output perdurant $e1.2\_outPD$. In other words, the perdurants that contribute to the achievement of $e1.2\_outPD$. Recall the relation *causes* defined in the Aspecto-FR ontology is very general, adopting the meaning of *causal contribution*. For that purpose this relation was defined as transitive. The column on the center shows a direct causal relation between the instance on the left and the instance on the right of the table. That is, the perdurants on the left stand on a *direct* causal relationship with the perdurants on the right, or more specifically, the perdurants on the left directly cause the perdurants on the right.

The relation *_d_causes* is a sub-property of *causes*, which is intended to be necessary cause for a change of state or for achieving a functional goal. This relationship was defined as *functional*, in the sense that it allows only one instance as its range, i.e. a m-to-1 relationship. Direct causes are used in the model to describe causality that is internal to the device or subsystem, i.e. device-centric causality set by design.

The next query selects all the perdurants the ventilation opening participates in. These

participation relations include the nominal participation asserted first, simple participation inferred by the reasoner, and all the causal participation relations also inferred from the modifications.

Listing 7.2: SPARQL query 2. Participation relations for ventilation opening a1.4.1.

```
SELECT ?x ?y ?z
WHERE
{  ?x owl:sameAs dev:a1.4.1 . # ?x is a technical artifact. From DOLCE-FR.
   ?y a owl:ObjectProperty.
   ?z a fr:EPD .
   ?x ?y ?z .}
```

Table 17 presents the results. The first three rows show the three participation relations inferred in scenario 1, before any change in the parameter values was made. The other six relations were added by the reasoner according to the conditions specified for scenario 4. This implements the query model discussed in the last Chapter, as part of the four models of the proposed modeling framework. Furthermore, this query demonstrates the second criteria of the research hypothesis (5.2) regarding multiple participation of endurants in multiple perdurants, as condition for the identification of multi-functionality and potential functional conflicts across different systems.

Table 17: Query results for participation relations for part a1.4.1.

dev:e1.4.1	dc:participant-in	dev:e1.3.4
dev:e1.4.1	fr:nominal_participant_in	dev:e1.3.4
dev:e1.4.1	fr:causal_participant_in	dev:e1.2_outPD
dev:e1.4.1	fr:causal_participant_in	dev:e1.3_outPD
dev:e1.4.1	fr:causal_participant_in	dev:e1_outPD
dev:e1.4.1	fr:causal_participant_in	dev:e1.3.2
dev:e1.4.1	fr:causal_participant_in	dev:e1.3.3
dev:e1.4.1	fr:causal_participant_in	dev:e1.2.1
dev:e1.4.1	fr:causal_participant_in	dev:e1.3.1

Finally, the second criteria of the research hypothesis can be verified with the next query, dealing with the participants in an intended output perdurant. In other words, this query provides the Aspect System of a function, and together with the previous one, allows the identification of cross-cutting functional interactions at multiple levels of abstraction. In

the query, the variable ?x stands for a participation relation. Hence, the query returns all artifacts which are co-participant in the achievement of e1.2_outPD.

Listing 7.3: SPARQL query 3. Aspect system for $e1.2\_outPD$.

```
SELECT ?x ?y
WHERE
{ ?x a owl:ObjectProperty.
  ?y a fr:Technical_artifact.
  dev:e1.2_outPD ?x ?y .
  FILTER(?x != dc:immediate-relation)}
```

The fact that an aspect system for a function has been inferred from a device-centric perspective is because an environment has been defined, containing only air and the device. No other external effect or participant is involved, based on the conditions asserted for the model. Another modeling option would be to assert the device as the environment itself, but then the role of air in ventilation would have to be omitted.

Table 18: Aspect system of sub-function $e1.2\_outPD$, described as output perdurant.

Scenario 1	**Scenario 2**	**Scenario 3**	**Scenario 4**
dev:a1	dev:a1	dev:a1	dev:a1
dev:e1.2	dev:a1.2	dev:a1.2	dev:a1.2
-	dev:a1.4	dev:a1.4	dev:a1.3
-	-	dev:a1.4.1	dev:a1.4
-	-	-	dev:a1.4.1

7.2 Case study 2: Ballasted PV racking system - Part A

This case was also introduced earlier in Chapter I, to exemplify design situations where the structural composition of assemblies are changed by the removal of parts. More specifically, the example illustrates the scenarios where an intended functionality of a multi-functional component is inadvertently removed from the system, as result of a change in the material properties, or when the component is eliminated altogether. Hence, the main interaction patterns being to be captured are not those derived from simple dimensional changes as illustrated in the first case study, but have to deal with more meaningful modifications of the composition hierarchy, involving specification of structural constraints at the topological level, as well as behavioral constraint at the functional level.

The design of a ballasted photovoltaic (PV) racking system for flat roof buildings exemplifies the general problem of this research. In particular, it allows to the test the prototype implementation regarding its ability to support multiple functional views at different levels of abstraction, which are required to validate the criteria of functional integration and multi-functionality that constitute the main research hypothesis. In the particular design used for this case study, the criteria spans several stages of a product life-cycle. This is because one of the main guiding principles behind the design project was the development of multi-functional parts, as general strategy to drastically reduce manufacturing and installation costs associated with the hardware used in solar systems[2].

Among the several multi-functional components developed for this product, the wind deflector/wind skirt stands out as special sub-case. Because this was the largest part of the entire assembly, using a significant amount of steel, it was considered early on in the design process that the only way of justifying its cost was by adding extra functionality. Therefore, besides keeping its 'nominal' function of deflecting wind loads to keep the solar array in position, the wind deflector was designed to accommodate an additional set of integrated functions, including:

1. Packaging and packing, for storage and transportation.

[2]The author of this research was also one of the lead designers of this product.

2. Pre-squaring, to facilitate assembly without measuring tools.

3. Supporting base rails, to facilitate attachment of PV modules.

4. Wire management, to support and cover bundles of electrical wire.

5. Bracing the array under lateral loads.

The pre-squaring function allows the installation crew to set the base rails parallel to each other, positioning them at the correct distance for the attachment of the PV modules. This is done by fastening the extremes of the wind deflector directly to the top mounting supports of the rails (triangular portion in Figure 7.10). In this way the base rails and mounting supports stay rigid in a vertical position to receive PV modules on top later. Without that functionality, the process would require the use of measuring tapes and chalk lines to define the geometry of the array. Figure 7.2 illustrates some of these functions during the deployment of the system.

During later phases of design, the elimination of the wind-deflector was considered an option to reduce manufacturing costs even lower. To compensate for uplift wind loads, the sectional area of base rails was increased. Other additional functions, such as packaging and wire-management were sacrificed, given the new cost benefits. However, the pre-squaring function was completely overlooked during the decision-making process. This was one of most important features of the system, contributing significantly to the reduction of installation time and cost. Despite its relevance, the designers could not recall the negative implications until much later.

Figure 7.9 describes the main components of the racking assembly, which were modeled logically as part of the initial asserted model. Figure 7.10 describes the composition structure of the assembly, with the labels used in the logical model. Recall that all structural information is intended to be provided in the future directly from the CAD model.

Figure 7.8: Multi-functional wind-deflector design. Top left: Packaging and packing function. Top right: integrated squaring and integrated wire management function. Bottom: squaring of base rails prior attachment of PV modules.

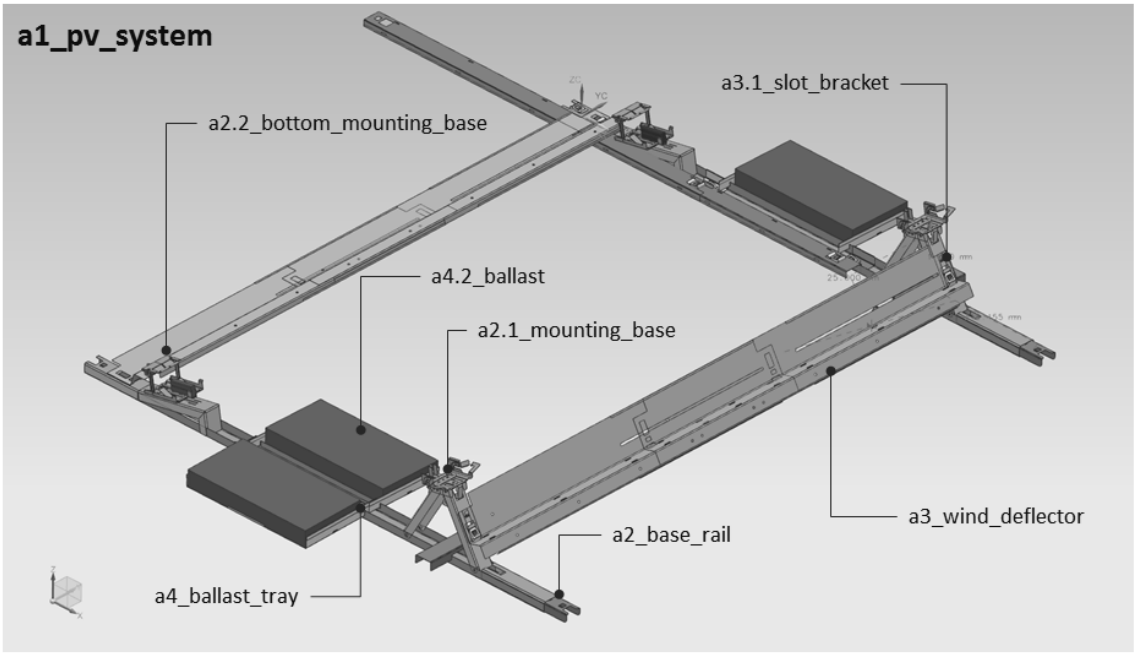

Figure 7.9: Anaconda composition hierarchy

7.2.1 Identified task: design scenarios

The functions described above were describe in the functional model, along with the logical representation of the design structure. The two main design scenarios involve the comparison of total manufacturing cost for the entire racking assembly (for just one PV module) with and without the wind deflector. However, the objective is to identify the functional implications of removing the wind deflector from the design by means of a series of queries regarding the aspect systems of different functions. This requires the specification of functions beyond simple participation, as done in the first case study. More specifically, the problem requires the specification of structural constraints related to connection between parts, based on OWL-DL characteristics such as symmetry and qualified cardinality restrictions.

Besides modeling the wind deflector functions described above, the model developed for this case study also includes the sub-function of attachment between the wind deflector and the mounting supports of the base rails (the triangular part in Figure 7.9, labelled $a2.1$). This attachment function could be provided by two options, developed in the model to

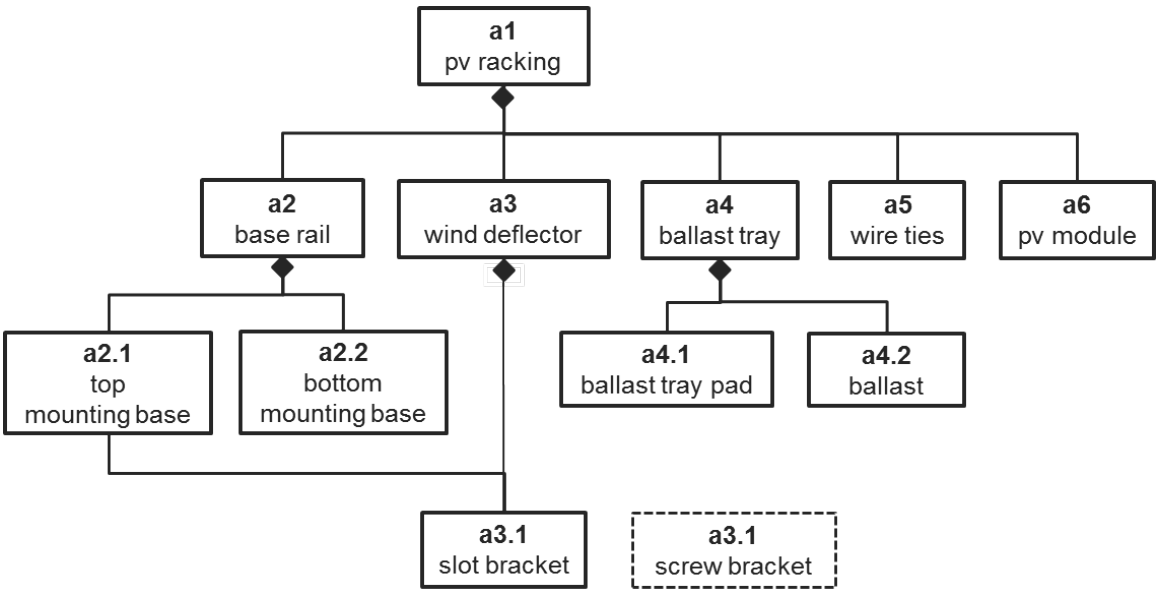

Figure 7.10: PV racking composition hierarchy

illustrate the functional implications of design changes across various levels of abstraction.

The first attachment option is based on the use of a screw bracket ($a3.1\_screw\_bracket$), which provides a rigid connection between the wind deflector $a3$ and the mounting support $a2.1$, but requires the use of a screw driver and additional parts, increasing the number of steps and overall time of installation. The second option is the use of a slot bracket ($a3.1\_slot\_bracket$), which relies on fitting a slot hole on each side of the wind deflector in a corresponding hook of both mounting supports. This second option does not requires tools, and it is simpler and faster, but provides limited bracing capabilities under lateral load conditions such as seismic and wind loads. Furthermore, all metal parts of the system have to be electrically grounded, which can be done with special screws or star washers, but not with simple slots and hooks. Therefore, each option offers different capabilities and advantages, which need to be captured by a different aspect system, and according to the levels of performance required for the same function.

7.2.2 Problem formalization and encoding

This model consists of assertions about classes, properties and rules, as well as assertions of facts about individuals at the instance level. The following subsections describe the most

relevant ones, first addressing functionality from a device-centric viewpoint, and then from an environment centric view-point. All assertions made in OWL-DL are presented according to the Manchester syntax. The prefixes of source ontologies used are included in the listings for the assertion of individuals. For example, the prefix for the Aspecto-FR ontology is fr:, and for the PV racking system ontology is sol:.

7.2.2.1 Device-centric assertions

The following listing shows the assertions for the fastening functions of a slot bracket and a screw bracket. These functions are defined from a device-centric perspective, as denoted by the use of '_fd' at the beginning of their names, where the 'd' stands for 'device'. This convention is for ease of readability only. The indices _fd2.1 indicate that these are invoked as sub-functions of the pre-squaring function _fd2_pre−squaring_D_function, introduced later.

Listing 7.4: Device-centric specification of fastening function for two bracket types.

```
1  Individual: sol:_fd2.1 _slot_fastening_D_function
2     Facts:
3        fr:_DF_artifact   sol:a3.1 _slot_bracket ,
4        fr:b0   sol: b_slotting_1 ,
5        fr:b1   sol: b_slotted
6  Individual: sol:_fd2.1_screw_bracket_D_function
7     Facts:
8        fr:_DF_artifact   solar:a3.1_screw_bracket,
9        fr:b0   solar:b_screwing,
10       fr:b1   solar:b_screwed
```

The facts asserted follow the specification of a device-centric functional perspective defined in DOLCE-FR as $DevFunc_G(a, b_0, b_1)$, where the term 'Aspecto−FR:_DF_artifact' corresponds to the artifact a, while Aspecto−FR:b0 and Aspecto−FR:b1 are the behaviors b_0 and b_1, respectively (see 23). Each of these behaviors qualifies the participation of the brackets in the corresponding input and output perdurants, allowing the specification of behavioral constraints over such participation. Assertions for these behaviors are:

Listing 7.5: Asserted behavioral constraints on *e_screwing_3*.

```
Individual: sol:b_screwing
    Types:
        sol : Fastening_capability
    Facts:
        fr : behavior_constraint_on_perdurant   sol : e_screwing_3
Individual: sol:b_screwed
    Types:
        sol : Fastening_capability
    Facts:
        fr : behavior_constraint_on_perdurant   sol : e_screwed
```

Thus, the behaviors b_screwing and b_screwed qualify the participation of the screw bracket in the perdurants e_screwing and e_screwed respectively, where the latter is the intended output perdurant of the function. The names of perdurants start with 'e_'. This schema for a device function establishes an objectified relation between the artifact and an intended output behavior. In this example, the function above assigns a fastening capability to the screw bracket. Internally however, a nominal participation is being qualified between the screw bracket and the *event of getting the bracket screwed*, which is precisely the intended output perdurant of this device function. Because of these relations, the entity _fd2.1_screw_bracket_function does not need to be asserted as a device function (type _D_Function in the Aspecto-FR ontology). Instead, it is inferred automatically as such, given proper relationships established with other entity types.

Is important to point out that this device-centric functional perspective shares some similarities with the basic SBF schema for a function discussed in Chapter IV (4.2.1. In particular b0 and b1 are specifications of preconditions (e.g. GIVEN) and post-conditions (i.e. MAKES) that need to be satisfied. However, the internal causal mechanisms that accomplish the function are not referred by a pointer, as in SBF (i.e. BY-BEHAVIOR), but are implicit in the set of preconditions themselves specified by b0. This is because the data structure used to build causal chains of perdurants is somewhat analogous to a linked

list. The structure of this causal chain was asserted in forward manner, as showed in the listing below. However, the inverse causal relations are inferred automatically.

Listing 7.6: Causal chain to achieve a *e_screwed* goal state.

```
1 Individual: sol:e_screwing_1
2   Facts:
3     fr : causes_directly   sol : e_screwing_2
4 Individual: sol:e_screwing_2
5   Facts:
6     fr : caused_directly_by   sol : e_screwing_3
7 Individual: sol:e_screwing_3
8   Facts:
9     fr : causes_directly   sol : e_screwed
```

Here, the intended output perdurant e_screwed acts like the 'head' of the linked list, while e_screwing_3 act like the tail or body of a linked list. Therefore, the behaviors b1 and b0 are used pointers to the head and body respectively. Because each perdurant in the body can have multiple causes, or may cause other perdurants besides the next one in the asserted chain, this data structure cannot be considered a proper linked list.

From a domain perspective, the causal chain above describes in a succinct manner the three main steps of fastening with a screw. The first step is the perdurant e_screwing_1, which refers to the alignment of mating holes for the parts being fastened. The second step e_screwing_2 is the pre-threading of the screw (or bolt) into the expose fastening hole. The third step e_screwing_3 is the screwing itself. This achieves the output perdurant e_screwed.

In any case, it is important to verify the internal conditions of satisfiability for the behavioral constraints that characterize a function. This means that there must be a causal chain effectively linking the perdurants indicated by b1 and b0. This verification is done by using a defined class in the Aspecto-FR core ontology.

A defined class, also known as equivalent class, relies on axioms of equivalence for the specification of membership conditions. This is often used as a form of query in OWL-DL. In Aspecto-FR, this defined class is called a Satisfiable_D_Function, which is presented below

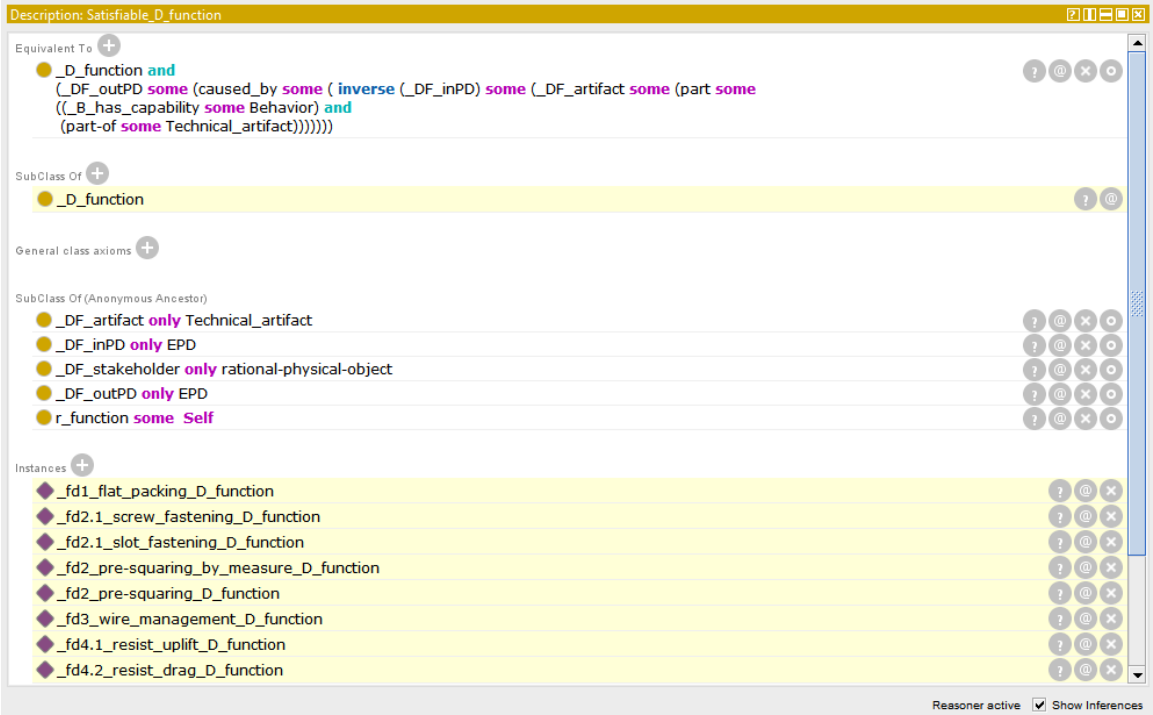

Figure 7.11: Defined class for Satisfiable_D_Function. Equivalence axiom at the top. Inference of individual members at the bottom (purple diamonds).

according to the Manchester syntax.

Listing 7.7: Equivalence axiom for a Satisfiable_D_function. Prefixes omitted.

```
1 Class: Satisfiable_D_function
2   EquivalentTo:
3     _D_function
4     and (_DF_outPD some (_c_caused_by some (inverse (_DF_inPD) some (_DF_artifact some
5           (dc:part some ((_B_has_capability some Behavior)
6     and (part-of some Technical_artifact)))))))
```

Informally, this equivalence axiom means that a *satisfiable device function* is any device function for which the specified output perdurant is linked to an input perdurant by an actual causal chain; and that such device function has been allocated to an artifact with the required behavioral capability (i.e. a behavioral constraint). Figure 7.11 shows the definition of the Satisfiable_D_Function class, along with its inferred members, in Protégé.

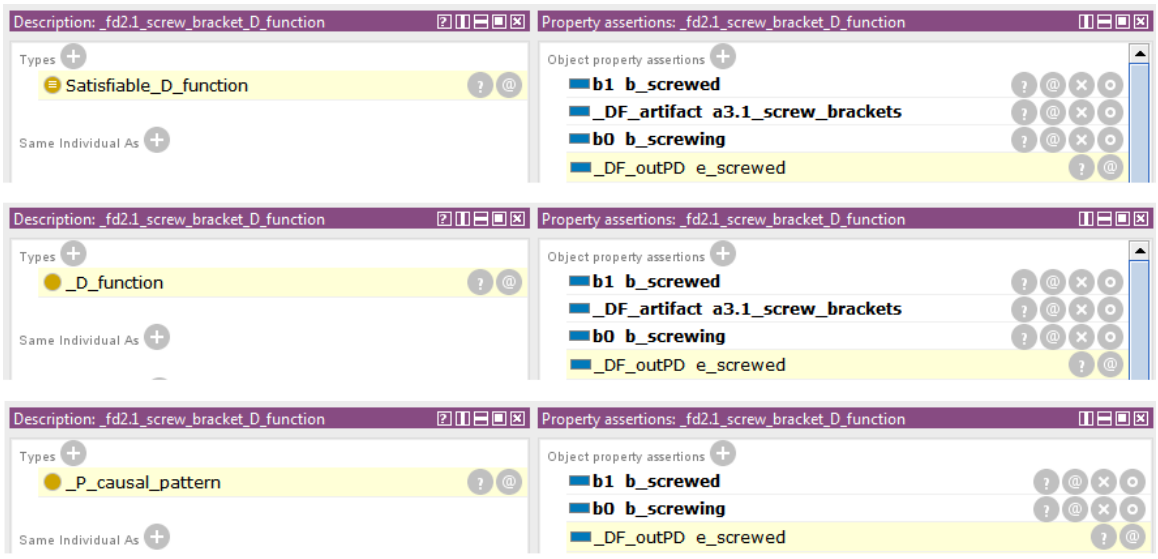

Figure 7.12: A satisfiable device function (top). A device function that is no longer satisfiable due to a broken or missing causal chain (center). A function with causal chain but no allocation to artifact classified as a generic causal pattern.

The schema proposed in Aspecto-FR for the definition of device-centric functions allows a modular, incremental specification. This means that functions can be defined independently of any particular artifact, or it can be allocated to an artifact without specification of any participation relation. That is, behavioral constraints can be associated to the function, but no participation relation may exist with a perdurant. Furthermore, even if participation relations are established, the underlying causal chains may not be set properly.

Figure 7.12 exemplifies these situations, first when the causal chain for the device function _fd2.1_screw_bracket_D_function is broken or missing. As result, this entity is no longer inferred as a Satisfiable_D_function, but only as a _D_function. In the second case, the function does have a valid causal chain but it is not allocated to any artifact. In this situation the function is classified by the reasoner as an abstract causal pattern (_P_causal_pattern in Aspecto-FR).

Another type of assertions done in the model deals with participation of endurants in perdurants of the causal chain. Such participation might be a nominal one, which denotes a sub-function or a functionality that is external to the device, or it might denote another form of participation, such as those resulting from a side-effect. For example, the process of

screwing has screws and a screw driver as nominal participants. This is asserted according to the following statements:

Listing 7.8: Asserted nominal participants in *e_screwing_3*.

```
1 Individual: sol:a8_screw_driver
2     Facts:
3         fr:nominal_participant_in  sol:e_screwing_3
4 Individual: sol:a8_screw_set
5     Facts:
6         fr:nominal_participant_in  sol:e_screwing_3
```

A similar set of assertions is done regarding an alternative fastening mechanism based on slot brackets. However, in this case no screws, bolts or tools are necessary, and the number of installation steps will be smaller as result. This consequences will be queried from an environment-centric perspective, since they depend of other conditions from the larger operational context of the installation.

Before being able to reach that perspective, it is necessary to compose low level functions into intermediate ones, such as the slot fastening function into the squaring function. The following listing shows the composition of these functions. Notice that the input behavior b0 for the pre-squaring function refers to the sub-function _fd2.1 _slot_fastening_D_function, instead of a normal input behavior. This feature allows for a recursive structure in the composition of device functions. In this way, the sub-function as a whole qualifies the participation of the wind deflector in the fastening perdurant.

Listing 7.9: Device-centric squaring function, composed by sub-function slot fastening.

```
1 Individual: sol:_fd2_pre-squaring_D_function
2     Facts:
3         fr:_DF_artifact  sol:a3_wind_deflector,
4         fr:b0 sol:_fd2.1 _slot_fastening_D_function ,
5         fr:b1 sol:b_integrated_pre-squaring
```

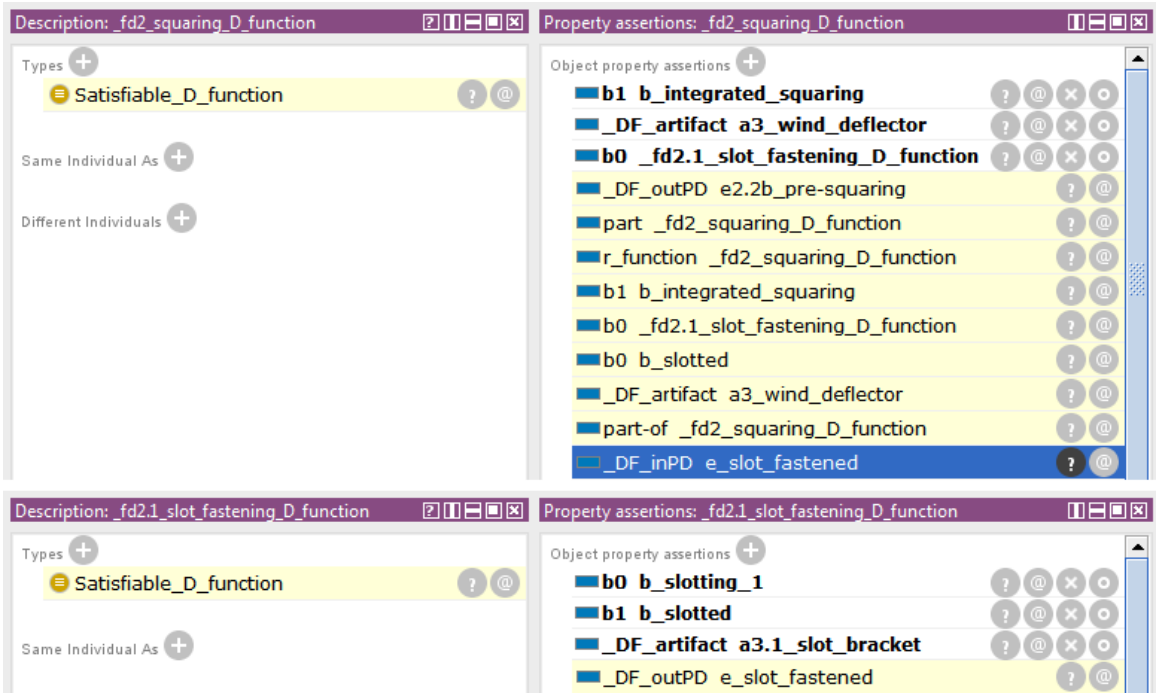

Figure 7.13: Recursivity of device functions with satisfiable conditions. Input behavior b0 of pre-squaring function of wind deflector (at the top) invokes fastening sub-function of the slot bracket (at the bottom).

Figure 7.13 shows the pre-squaring function _fd2 aggregating the slot fastening sub-function _fd2.1 as its input behavior.

Internally, an auxiliary SWRL rule sets the proper relationships between the output perdurant _DF_outPD and the input perdurant _DF_outPD of the pre-squaring function only if the sub-function _fd2.1 is satisfiable. This rule is defined as follows:

$$b0(?f, ?sf) \wedge \_P\_causal\_pattern(?f) \wedge b1(?sf, ?b1) \wedge$$

$$Satisfiable\_D\_function(?sf) \rightarrow b0(?f, ?b1)$$

Whenever the internal device sub-function is not satisfiable, either because it has no causal model defined, or because an artifact has not been allocated, then the device function invoking the sub-function is not satisfiable either. In the case of the pre-squaring function of the wind deflector, this means that it lacks a part or feature capable of being fastened to something else, and therefore, the wind deflector itself lacks such capability, which in turn

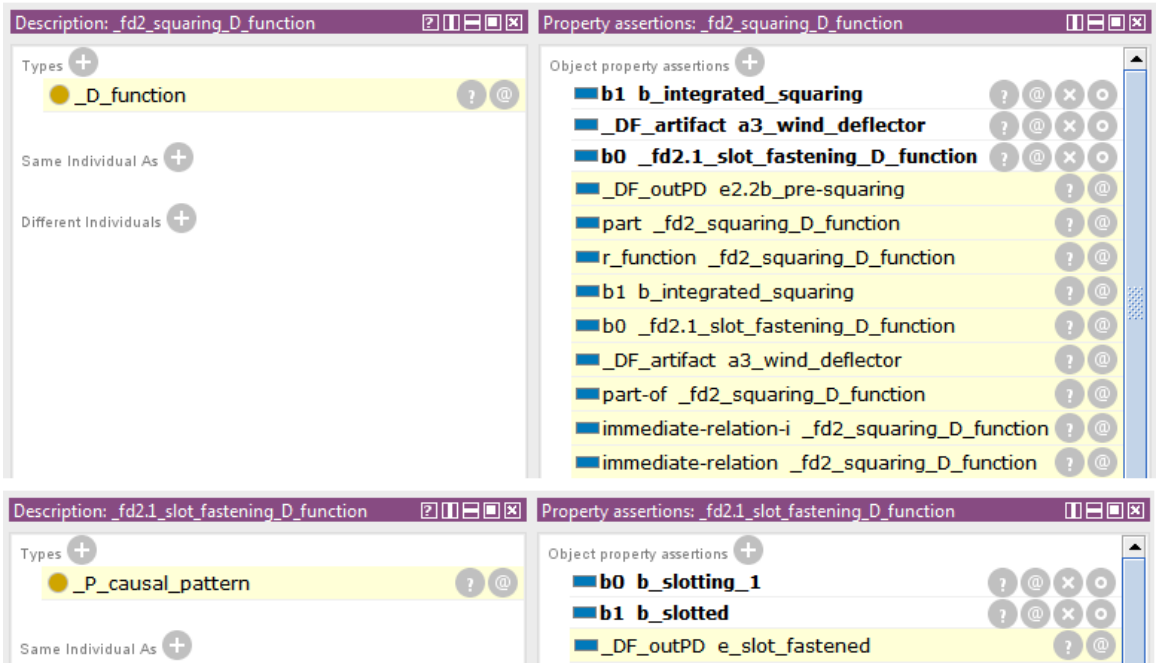

Figure 7.14: Recursivity of device functions with no satisfiable conditions. Pre-squaring function allocated to wind deflector (at the top) is no longer satisfiable because the fastening sub-function has not been allocated to any artifact (at the bottom).

does not allow it perform as a pre-squaring device.

Figure 7.14 shows an example in which the sub-function _fd2.1 _slot_fastening_D_function has not being allocated to a bracket (on the bottom). This renders the function not only as not satisfiable, but it also 'downgrades' the function to a simple causal pattern. As result, the function _fd2_pre−squaring_D_function is no longer a satisfiable _D_function.

The fact that an device function is not satisfiable just means, in this context, that it lacking information to be a well-formed, modular and reusable unit. The intended output perdurant being referred by the function can still be satisfied given the right environmental conditions. For instance, the wind deflector could still be used as squaring jig using quick-release clamps for temporary attachment to the base rails, or duct tape. Clearly, these two options are far from ideal, but they serve to illustrate the point.

With these definitions and assertions in place, now it is possible to address the functionality of the PV racking system, and the wind deflector in particular, from an environment-centric perspective.

7.2.2.2 Environment-centric assertions

Since the goal of the functional modeling framework proposed here is to support inference of functional interactions from an environment-centric perspective, but across levels of abstraction, a series of high-level functions related to the rooftop PV racking system have been defined.

These are divided in two main groups. The first deals with the installation process, which includes the installation of hardware and the electrical installation, and the second deals with the structural function of maintaining the form and position of the solar array during its usable life. The environment-centric perspective is denoted by the prefix _fe, instead of the _fd used previously.

- _fe1_whole_installation
  - _fe2_hardware_installation
    - _fe2.1_squaring
    - _fe2.2_p_module_attachment
  - _fe3_electrical_installation
    - _fe3.1_wire_management
- _fe5_maintain_form_position

Recall that this system is fixed on top of flat roofs by means of ballast only. Therefore, it should be designed to resist seismic and wind loads that could cause the entire array to move sideways, causing failure by buckling of structural members of the racking, including connections and the PV modules themselves. Other causes of failure could be buckling of linear members by thermal expansion as well as deflection by excessive wind uplift forces.

Similarly to device-centric functions, an environment-centric function can also be composed of sub-functions, both device-centric or environment-centric. Conditions of satisfiability also apply, but in a different, more complex way related to constraints that are relevant at the environment level. Before discussing these constraints, the initial assertion of the squaring function from an environment-centric perspective is made as follows:

Listing 7.10: Asserted environment-centric version of squaring function.

```
Individual: sol:_fe2.1_squaring_function
    Facts:
        fr:_EF_in_environment kb:Env1,
        fr:_EF_in_mode_of_deployment sol:_md2.1,
        fr:_EF_nominal_artifact  sol:a1_pv_racking,
        fr:b0  sol:b_some_pre−squaring,
        fr:b1  sol:b_squared
```

Notice that the output behavior b1 is b_squared, which is a constraint over the output perdurant e_squared. How this goal state is achieved is specified by the input behavior b0. In this case, by some open-ended squaring method, i.e. b_some_pre−squaring. Alternatively, a more specific behavior could be specified. For instance, by means of invoking the device-centric squaring function _fd2_pre−squaring_D_function itself as input behavior. This adds a more specific constraint on how the intended square state should be achieved. Such specification is asserted as follows:

Listing 7.11: Alternative function specification. Device-centric squaring function is invoked.

```
Individual: sol:_fe2.1_squaring_function
    Facts:
    ...
        fr:b0  sol:_fd2_pre−squaring_D_function,
        fr:b1  sol:b_squared
```

The output of _fd2_pre−squaring_D_function is a pre-squared state, not a fully squared state. This means that other events need to happen in the environment before the function could be fulfilled. These other events are precisely the *mode of deployment* of the environment-centric function, as formalized in DOLCE-FR and discussed in the hypothesis (Chapter 5, section 5.1.5). The implication is that a more complex causal model needs to be described, which includes new causal links, along with possible new participants from the environment. In any case, the resulting set of participants conform the aspect system

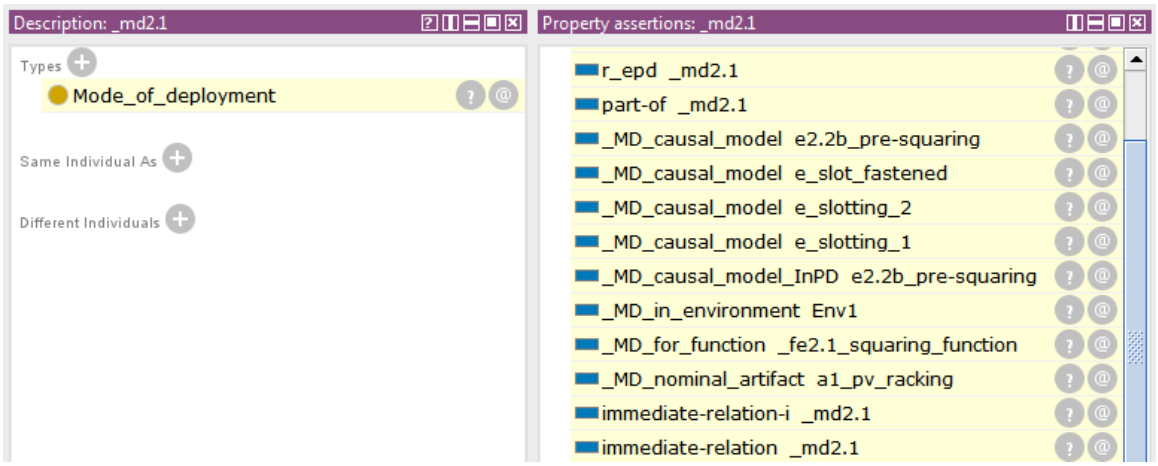

Figure 7.15: Mode of Deployment for squaring function. Causal links leading to output perdurant are inferred by backward chaining. These links are denoted by property _MD_casual_model.

of the squaring function.

An instance of the mode of deployment is referenced in specification of the environment-centric function itself. In the case of the squaring function, this is done by relating the function to the individual _md2.1 with the object property _EF_in_mode_of_deployment, as seen in the second fact asserted in the previous listing 7.25.

Mode of deployment _md2.1 captures the causal links that lead to the output perdurant e_squared, by means of backward chaining. Figure 7.15 illustrates how the output perdurant e_squared results from a series of events and states, most of which are enabled in this model by the fastening function of the slot bracket (e_slot_fastened), asserted as part of the wind deflector. The 'slotting' events are similar to the 'screwing' events declared for the causal chain leading to e_screwed presented in listing 7.6. Thus, e_slotting_1 causes e_slotting_2 , which finally causes e_slot_fastened .

It is important to note however that the mode of deployment of a function is a perdurant, or more precisely, a fusion of perdurants. Therefore, it does not contain information about the participants involved. That role belongs to the aspect system of the function.

The reference to the environment is done directly in the assertion of the function, as it can be seen in the first asserted fact for _fe2.1_squaring_function (in listing 7.25). This is

asserted with the object property _EF_in_environment relating the function with a specific environment instance, in this case Env1. The figure also shows that a relationship with the environment Env1 is inferred automatically for the mode of deployment _md2.1.

This relationship is key, because a mode of deployment is dependent on the composition of the environment. Therefore, given a different composition for Env1, a different mode of deployment, and by consequence, a different set of participants, or aspect system, would occur for the same function. To illustrate that dependency it is necessary to establish the relationship between an aspect system and its function.

There are two forms in which this can be done, as discussed in the previous chapter regarding the definition of the ontology and knowledge base models. The first involves the formulation of a defined class based on an equivalence axiom. For the case of the squaring function, that axiom is:

Listing 7.12: DL query for aspect system of squaring function.

```
dc:part some
    (dc:participant-in some (_c_causes some
      (inverse  fr:_EF_outPD value sol:_fe2.1_squaring_function)))
and (dc:proper-part-of value kb:Env1)
```

For the environment initially asserted, and the mode of deployment _md2.1 depicted in Figure 7.15, the inferred aspect system for the squaring function includes eleven members. Figure 7.16 shows the equivalence axiom, used in this example as a DL query. Inferred members are represented here as instances of the class expression representing the aspect system of the function. This also means that the function is *realizable* according to the conditions specified for the environment.

Inferred instances need to be a proper part of the environment, according to the definition of aspect systems provided in the previous chapter. However, it is often necessary to describe additional structural relationships taking place in the environment, along with different types of constraint, as discussed preliminarily in the first case study.

In this case, the environment-centric specification of the squaring function relies on the

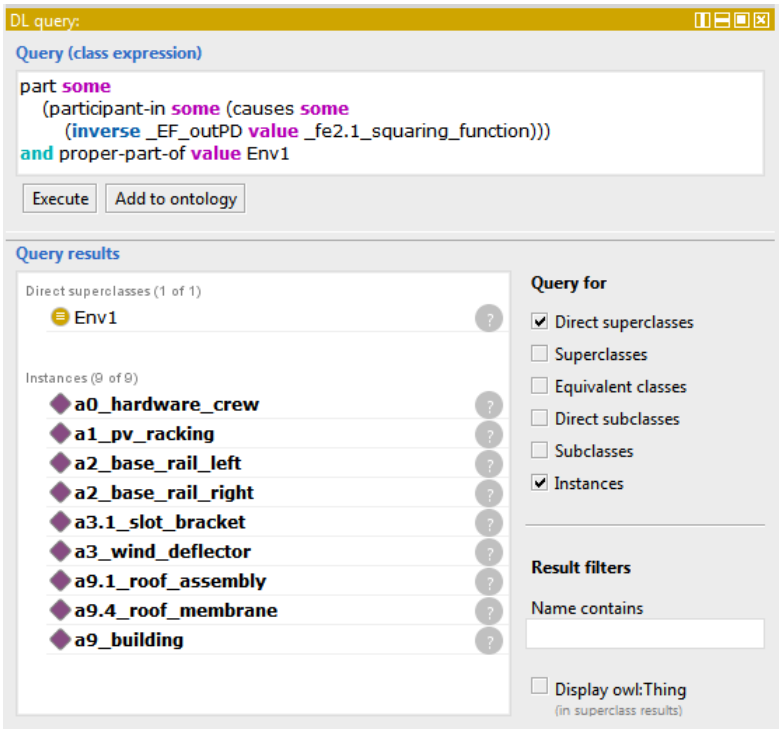

Figure 7.16: Aspect system of squaring function as defined class expression.

device-centric specification of squaring function allocated to the wind deflector (_fd2_pre−squaring_D_function), which in turn relies on the device function (_fd2.1 _slot_fastening_D_function) of the slot bracket a3.1 _slot_bracket . As it can be seen in Figure 7.16, this is correctly inferred by the reasoner with the addition of the wind deflector and slot bracket as part of the inferred instances.

For this to happen, two additional structural constraints are required. The first requirement is that the slot bracket needs to be declared as part of the wind deflector, since it provides the basic fastening capabilities. The second is that the wind deflector needs to be structurally connected to the rest of the racking system. More specifically, the wind deflector needs to be connected by means of a direct connection between the slot bracket and the mounting bases of the base rails.

The assertion about this part-of relation is presented first in the listing below. The direct connection between the slot bracket and the mounting based is asserted next. Manufacturing costs have been added as part of the facts asserted for each of these racking components, to be used by the queries developed later.

Listing 7.13: Assertion of structural composition and connectivity.

```
1  Individual:sol:a3_wind_deflector
2      Types:
3          sol:Wind_deflector
4      Facts:
5          dc:proper−part sol:a3.1 _slot_bracket ,
6          sol:has_manufacturing_cost 5.43,
7  Individual: sol:a3.1 _slot_bracket
8      Types:
9          sol:Bracket
10     Facts:
11         fr:s_directly_connected_to   sol:a2.1_mounting_base_left ,
12         fr:s_directly_connected_to   sol:a2.1_mounting_base_right,
13         sol:has_manufacturing_cost 0.1
```

The s_directly_connected_to relation is defined in the Aspecto-FR ontology as a functional object property, without transitivity[3]. Its super-property is s_connected_to, which is transitive. A similar modeling pattern in OWL has been used in the definition of part and proper-part relations, as well as in the definition of cause and direct cause relations that underpin the formulation of Aspecto-FR.

Altogether with the allocation of the fastening capability to the slot bracket (in listing 7.4), these constraints ensure that the squaring function is not only satisfiable, but actually realizable in the specified environment. The former means that a nominal artifact with the required capability for the function has been instantiated within the environment, and according to the constraints specified. In this example, the main constraint is physical connection. The latter ensures that a necessary causal model to accomplish the function is available. In the case of the squaring function, these conditions fail if:

- The bracket no longer has the fastening capability.
- The bracket is no longer part of the wind deflector.

[3]The term functional has a different meaning in OWL, limiting properties to a single have a single value.

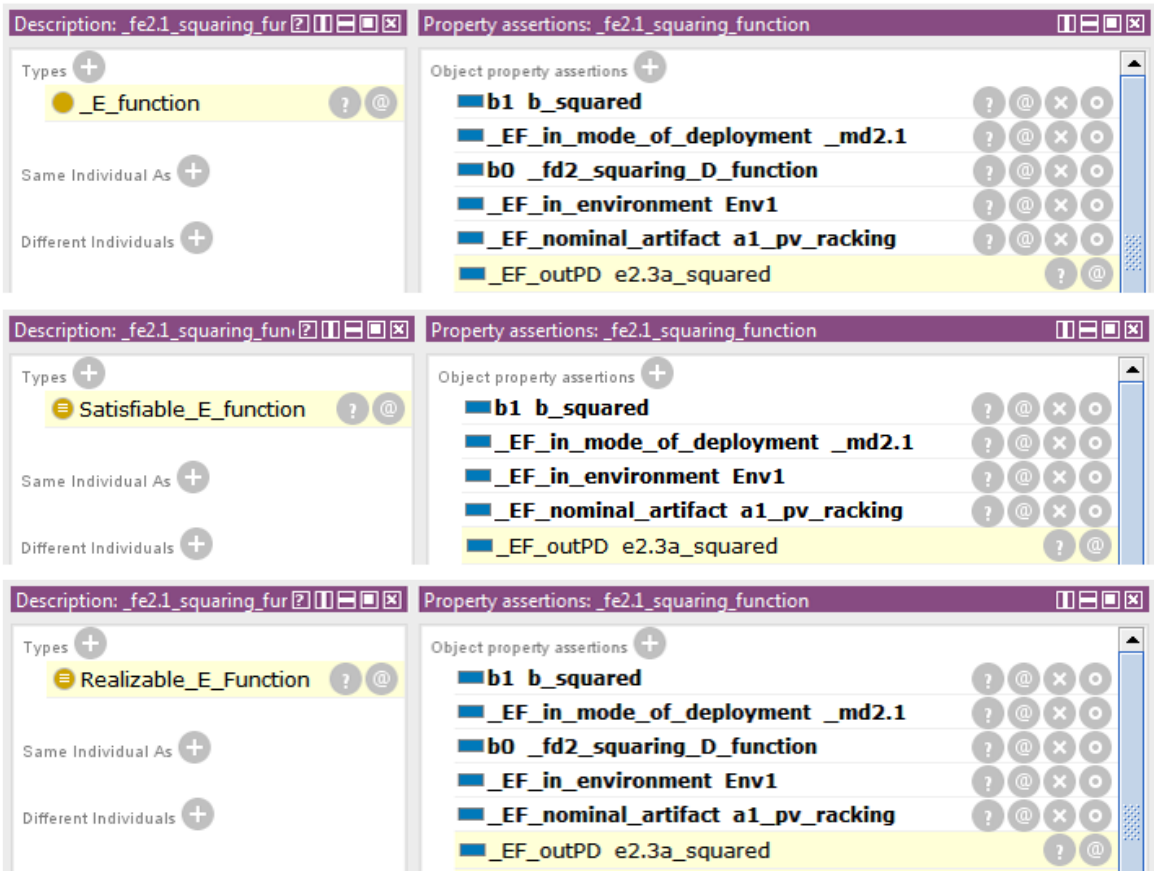

Figure 7.17: Automatic classification of realizable environment-centric functions.

- The bracket is no longer connected to the mounting base.

These different classification schemes are demonstrated in the Figure 7.17. On the top, an environment-centric specification for the squaring function that is not realizable nor satisfiable. This means that there might be a component in the environment with squaring capabilities required, but it is not physically connected. The connection constraint has been violated in this first scenario by removing the corresponding assertion regarding the slot bracket, made in the previous listing. Therefore the function is classified simply as an environment-centric function _E_function.

In the second scenario the function has been classified as a satisfiable environment-centric function (Satisfiable_E_function). This is because there is no complete causal chain available leading to the goal state e2.3a_squared, even if a component exists in the environment with the required capability. This classification is based on following equivalence class axiom:

Listing 7.14: Equivalence axiom for a *Satisfiable_E_Function*.

```
Class: fr:Satisfiable_E_function
    EquivalentTo:
        fr:_E_function
         and (fr:_DF_outPD some (_c_caused_by some (inverse (fr:_DF_inPD) some
         (fr:_EF_nominal_artifact some (dc:part some ((fr:s_connected_to some
          (dc:part−of value kb:Env1)) and (fr:_B_has_capability some fr:Behavior)))))))
```

The third scenario shows a complete, realizable environment-centric function (Satisfiable_R_function). This classification is made according to the following equivalence class axiom:

Listing 7.15: Equivalence axiom for a *Realizable_E_Function*.

```
fr:Satisfiable_E_function and (fr:EF_realizability value true)
```

Where EF_realizability is a Boolean data property that is inferred according to a SWRL rule that checks for a complete causal chain between the output and input perdurants. This rule works as an auxiliary function, and it is defined as according to:

$$\_c\_caused\_by(?o, ?i) \wedge \_EF\_inPD(?d, ?i) \wedge \_EF\_outPD(?d, ?o) \rightarrow EF\_realizability(?d, true)$$

If an environment-centric function is only satisfiable, then a possibly incomplete mode of deployment is being described, along with an incomplete aspect system for the function. If the function is not even satisfiable, then the aspect system has no members or parts. In this situation, the query presented in Figure 7.16 would not return instances at all.

As mentioned before, this would be the result of the function not having an artifact allocated with the required capability. In the case of the squaring function, this would be the function of either a component, such as the wind deflector, or an external set of squaring jigs and tools. Furthermore, it is not enough for a component to have the required capability; the component has to be connected to the system, along with other constraints, such that the mode of deployment required could be inferred as complete, and the function

rendered as realizable.

These conditions are set as part of the causal model of the squaring function, according to the equivalence axiom of the defined class Squared_E, described in the listing below:

Listing 7.16: Equivalence axiom for causal model of squaring function.

```
1  Class: sol:Squared_E
2      EquivalentTo:
3          sol:Squared
4           and (inverse (fr:_EF_outPD) some (fr:_EF_nominal_artifact some
5              ((fr:s_connected_to some (dc:part−of value sol:a1_pv_racking))
6              and (fr:_B_has_capability some sol:Squaring_capability)
7              and (dc:part−of some sol:PV_racking_assembly))))
8      SubClassOf:
9          fr:CauseMD,
10         fr:_d_caused_by value sol:e2.2b_pre−squaring
```

Informally, this axioms establishes the conditions under which the squaring function becomes realized. Similarly to the conditions of satisfiablility, it requires a part of the system to possess the required capability (e.g. a previously asserted device-centric function, such as pre-squaring), and to be connected to a part of the racking. If these conditions are met, then the output perdurant e_squared is causally linked to the causal model of the device-centric specification of the pre-squaring function. In particular, the output perdurant e_squared is inferred to be *directly caused by* the event e2.2b_pre−squaring, as specified in the last line of the listing.

This inference completes the mode of deployment for the environment-centric version of the squaring function. If the conditions are not met, then the mode of deployment of the function is incomplete, and by result, its aspect system as well.

This form of axiomatization allows to set constraints on how design elements have to be related with each other within a given environment. Any squaring device, such as the wind deflector, needs to be not only part of an environment, but also connected to a part of the racking, for the squaring function to be realized. This modeling pattern is used for

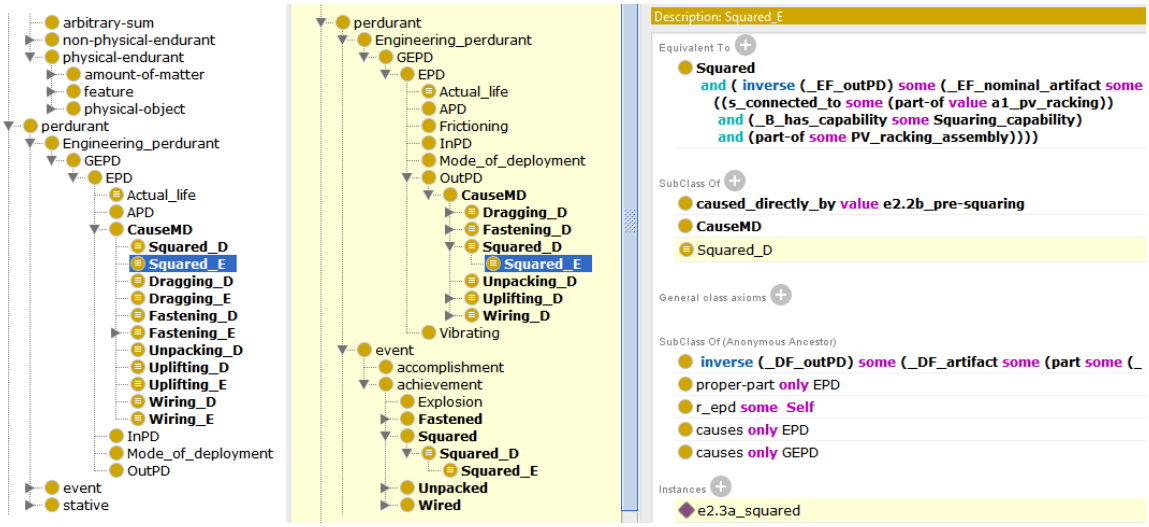

Figure 7.18: Equivalence axiom of causal model for achievement of squaring.

the inference of modes of deployment of other functions modeled in this case study, and explained according to the next design scenarios.

Figure 7.18 shows the result of the inference after running the reasoner. On the left column, the assertions of Squared_D and Squared_E as defined classes for completing the causal models of various functions. On the middle column, Squared_E is classified not only as subclass of Squared_D, but also as an output perdurant *OutPD*. On the bottom right, the individual e2.3_squared is inferred as an instance, thus rendering the squaring function realizable.

7.2.3 Design trade-off scenario 1

The first scenario is based on the initial design alternative for the PV racking system discussed in Chapter I. In this alternative, wind deflectors are used to reduce pressure differential between the top and bottom surfaces of PV modules, by redirecting wind loads that could lead to uplift and drag. This in turn reduces the need for additional ballast to maintain the form and position of the entire solar array, contributing to the structural integrity of the entire system.

To better understand the relationship between these requirements, it would be useful to revisit the difference between device-centric and environment-centric functional perspectives

in the context of the wind deflector. While the specific nominal function of this device is to deflect wind, the ultimate goal is to maintain the structural integrity of the solar array, while minimizing ballast. Therefore, wind deflection is only one of many possible solutions for the general problem. A similar observation can be made regarding the other device-centric, nominal functions, such as squaring or wire management.

Conversely, a device-centric function is usually not enough to ensure the satisfaction of a functional need at a higher level of abstraction, that is, from an environment-centric perspective. Often, additional functional roles and effects are required from other parts of the environment working together. To illustrate the difference, and how both perspectives complement each other, the following query returns the specification of all device-centric functions for which the wind deflector has a nominal participation. This is showed in the second column on the right. The first column on the left shows the intended effect or state specified by the function in terms of the intended output perdurant.

Listing 7.17: Query on device-centric function of wind deflector.

```
1 PREFIX sol: <http://www.ou.edu/coa/ontologies/2018/4/Aspecto-FR-Solar#>
2 SELECT (?x AS ?output_epd)(?y AS ?d_function)
3 WHERE
4 {  ?w a sol:Wind_deflector ;
5       fr:nominal_participant_in ?x .
6    ?y fr:_DF_outPD ?x .
7    ?y a fr:_D_function .}
```

The results of the query are presented in the table 19. As it can be seen, only four functions have been nominally allocated to the wind deflector. This does not mean that this device does not participate in the satisfaction of other functions, nor that it can satisfy by itself its nominal functions at the environment level.

Querying about the functions of the wind deflector from an environment-centric perspective returns a different set of output perdurants and functions. This query is presented in the listing below. The query also includes the direct cause for the intended output perdurant as an optional field. The returned results in table (20).

Table 19: Output perdurants and device-centric functions of wind deflector.

?output_epd	?d_function
sol:e1_flat_packing	sol:_fd1_flat_packing_function
sol:e2.2b_pre-squaring	sol:_fd2_pre-squaring_function
sol:e3.2_house_wire_bundles	sol:_fd3_wire_management_function
sol:e4.3_air_pressure_balance	sol:_fd4.1_resist_uplift_by_deflection_function

Table 20: Output perdurants and environment-centric functions of wind deflector.

?direct_cause	?output_epd	?e_function
-	sol:e1_flat_packing	sol:_fe0_self_packing
sol:e3.1_wiring	sol:e3.2_house_wire_bundles	sol:_fe3.1_wire_management
sol:e_wind_loading	sol:e4.1_uplift	sol:_fe6.1_maintain_position
-	sol:e4.3_air_pressure_balance	sol:_fe6.1_maintain_position
sol:e_wind_loading	sol:e4.1_uplift	sol:_fe6_maintain_structural_integrity
-	sol:e4.3_air_pressure_balance	sol:_fe6_maintain_structural_integrity

Listing 7.18: Query on environment-centric function of wind deflector.

```
SELECT DISTINCT (?x AS ?direct_cause)(?y AS ?output_epd)(?z AS ?e_function)
WHERE
  {  ?w a sol:Wind_deflector ;
        fr:nominal_participant_in ?y .
     ?z fr:_EF_outPD ?y .
     ?z a fr:_E_function .
     OPTIONAL{?y fr:_d_caused_by ?x} }
```

In particular, this query allows to visualize the difference between levels of abstraction for the functions of the wind deflector. Thus, the second query includes effects external to the wind deflector itself, which also play a causal role in the satisfaction of the environment-centric functions of position and structural integrity of the solar array. Specifically, the query returns (e_wind_loading) as playing a causal role in uplift (e4.1 _uplift).

Notice that the pre-squaring event e2.2b_pre−squaring is not the output of an environment-centric function, but of the device-centric function allocated to the wind deflector. At a higher functional level, the goal is to be achieved is the 'squared' state, denoted by the output perdurant e2.3a_squared. In this context, the pre-squaring function is a necessary but not sufficient pre-condition. The following query returns the causal link between the pre-squaring function of the wind deflector and the squaring function to be achieved from an environment-centric perspective.

Listing 7.19: Query on causal relation between device and environment-centric functions.

```
SELECT DISTINCT (?x AS ?causal_link)(?y AS ?output_epd)(?z AS ?e_function)
WHERE
{  ?w a sol:Wind_deflector ;
      fr:nominal_participant_in ?x.
   ?z fr:_EF_outPD ?y .
   ?x fr:_c_causes ?y .
   FILTER(?z = sol:_fe2.1_squaring_function)}
```

The results are showed in table 21. The left column shows the two nominal participation relations of the wind deflector, allocated as part of its device-centric functionality. As it can be seen in the table, e2.2b_pre_squaring is a direct cause to the output perdurant e2.3 a_squared, specified by the environment-centric function _fe2.1_squaring_function.

Table 21: Causal relation between device and environment-centric participation.

?causal_link	?output_epd	?e_function
sol:e2.2b_pre-squaring	sol:e2.3a_squared	sol:_fe2.1_squaring_function
sol:e1_unpacking	sol:e2.3a_squared	sol:_fe2.1_squaring_function

This query can be enhanced by qualifying the participation of the wind deflector in the perdurants involved. This is done by reference to the behavioral capabilities of the wind deflector. A behavioral capability denotes an inverse relationship to the behavioral constraints that apply during the allocation device-centric functions.

Listing 7.20: Query on behavioral capabilities / constraints of wind deflector.

```
SELECT DISTINCT
(?x AS ?causal_link)(?y AS ?output_epd)(?z AS ?e_function)(?b AS ?by_behavior)
WHERE
{  ?w a sol:Wind_deflector ;
      fr:_B_has_capability ?b .
   ?b  fr:behavior_constraint_on_perdurant ?x .
   ?z fr:_EF_outPD ?y .
   ?x fr:_c_causes ?y .
   FILTER(?z = sol:_fe2.1_squaring_function)}
```

In the three results returned (table 22), the participation of the wind deflector is qualified

by a description of the behavioral capability involved in the satisfaction of the environment-centric function _fe2.1_squaring_function. The behavioral capability is showed in the last column on the right. The prefix sol: has been removed to save space.

Table 22: Nominal participation for wind deflector with behavioral qualification.

?causal_link	?output	?function	?by_behavior
e_slot_fastened	e2.3a_squared	_fe2.1_squaring_function	b_quick_fastening
e2.2b_pre-squaring	e2.3a_squared	_fe2.1_squaring_function	b_integrated_squaring

Notice that the behavioral capability of the slot bracket is also returned as part of the query results. This is because the slot bracket is not only part of the wind deflector, but it is also the component that provides the b_easy_fastening capability of the wind deflector. In this way the wind deflector can be attached to the base rails while at the same time setting their correct position. If the bracket is removed from the wind deflector, such capability is lost and the wind deflector no longer can provide its integrated squaring capability.

Notice also that the other nominal participation of the wind deflector e1_flat_packing is included too, as result of being a necessary event preceding squaring. The behavioral capability that it provides is quick unpackaging, as opposed to the normal scenario where cardboard, metal straps and tape have to be removed and disposed by the installation crew.

However, this is not the entire set of nominal participation relations allocated to the wind deflector by the initial assertions. Other nominal participation relations have been made directly, without explicit reference to a functional specification. All nominal participations asserted to the wind deflector can be queried according to the next listing:

Listing 7.21: Query on all nominal participation relations of wind deflector.

```
1 SELECT (?x AS ?artifact)(?y AS ?nominal_participant_in)(?z AS ?by_behavior)
2 WHERE
3  {    ?x a sol:Wind_deflector ;
4       fr:_B_has_capability ?z .
5    ?z fr:behavioral_constraint_on_perdurant ?y}
```

The six nominal participation relations returned by the query are presented in the table below. It shows how the wind deflector has a nominal participation in supporting or

housing wire bundles, which is part of the wire management function. It also has a nominal participation in resisting wind uplift, part of the structural integrity function of the PV racking.

Table 23: All asserted nominal participations of wind deflector.

?artifact	?nominal_participant_in	?by_behavior
a3_wind_deflector	e1_hardware_unpacking	b_flat_unpacking
a3_wind_deflector	e_slot_fastened	b_quick_fastening
a3_wind_deflector	e3.2_wire_management	b_integrated_wire_support
a3_wind_deflector	e2.2b_pre-squaring	b_integrated_squaring
a3_wind_deflector	e4.1_uplift	b_reduces_uplift_by_deflection
a3_wind_deflector	e4.3_air_pressure_balance	b_keep_air_pressure_balance

Finally, the following query returns all the functions that the wind deflector participates in, including those in which the participation is the result of an unintended side-effect. This query corresponds to the second criterion of multi-functionality of the research hypothesis.

Listing 7.22: Query on all asserted and inferred participation relations of wind deflector.

```
SELECT DISTINCT
(?x AS ?causal_link)(?y AS ?output_epd)(?z AS ?e_function)(?b AS ?by_behavior)
WHERE
{  ?w a sol:Wind_deflector ;
      dc:proper-part-of kb:Env1 ;
      fr:causal_participant_in_u ?y .
   ?z fr:_EF_outPD ?y .
   OPTIONAL { ?w fr:_B_has_capability ?b .
              ?b fr:behavior_constraint_on_perdurant ?x .
              ?x fr:_c_causes ?y}}
```

The table 24 shows the query results, with both asserted and inferred participation relations of the wind deflector. Nominal participation refers to the participation explicitly allocated to the artifact by means of an assertion. It may or not have a behavioral qualification or constraint associated.

Other types of participation may be secondary, or a reference to a known side effect. The inference of causal participation in output perdurants denotes a functional interaction. For instance, since the wind deflector is made of steel, it has participation in electrical

conductivity, and as consequence, in electrical grounding, which is one of the required perdurants of the environment-centric function of electrical safety. No particular causal link has been asserted or inferred (as indicated by the '-' line), nor a specific behavior.

There is also a participation in e4_maintain_form, the output perdurant of _fe4_maintain_form , which is a sub-function of _fe4_maintain_structural_integrity (in the third and fourth rows). However, there is no causal link returned from this query. For that purpose, a different SPARQL expression is needed to show a more complete causal chain. This will be reviewed in design scenario 3 of this case study.

Table 24: All participation relations of wind deflector, including some behaviors.

?causal_link	?output_epd	?e_function	?by_behavior
-	e1_flat_packing	_fe0_self_packing	-
-	e3.2_house_wire_bundles	_fe3.1_wire_management	-
-	e4_maintain_form	_fe6.1_maintain_form	-
-	e4_maintain_form	_fe6_maintain_structural_integrity	-
e1_flat_packing	e2.3a_squared	_fe2.1_squaring_function	b_quick_unpackaging
e2.2b_pre-squaring	e2.3a_squared	_fe2.1_squaring_function	b_integrated_squaring
e_slot_fastened	e2.3a_squared	_fe2.1_squaring_function	b_quick_fastening
e4.3_air_pressure_balance	e4.2_drag	_fe6.1_maintain_position	b_keep_air_pressure_balance
e4.3_air_pressure_balance	e4.2_drag	_fe6_maintain_structural_integrity	b_keep_air_pressure_balance
e3.2_house_wire_bundles	e3.4_electrical_system_tested	_fe5_electrical_installation	b_integrated_wire_management
e3.2_house_wire_bundles	e3.4_electrical_system_tested	_fe1_whole_installation	b_integrated_wire_management
e4.3_air_pressure_balance	e4.1_uplift	_fe6.1_maintain_position	b_keep_air_pressure_balance
e4.3_air_pressure_balance	e4.1_uplift	_fe6_maintain_structural_integrity	b_keep_air_pressure_balance
-	e4.3_air_pressure_balance	_fe6.1_maintain_position	-
-	e4.3_air_pressure_balance	_fe6_maintain_structural_integrity	-
-	e_electrical_grounding	_fe5.2_electrical_grounding	-

7.2.4 Design trade-off scenario 2

In the second scenario, the wind deflector is eliminated from the design to reduce manufacturing costs. For that purpose, it would be useful to have a functional model in which queries associated with costs can be made in relation with the functional implications of the proposed design changes. To illustrate the situation, the first query below returns the manufacturing cost for each the fourteen parts of the racking system considered in the first design alternative described in scenario 1. These parts are enough to support only one PV module, which is not included.

Listing 7.23: Query on manufacturing cost and weight per unit. Design scenario 1.

```
SELECT (?x AS ?part_unit)(?y AS ?unit_cost) (?z AS ?unit_weight)
WHERE
{  ?a rdf:type sol:PV_racking_assembly ;
      dc:proper-part ?x .
   ?x sol:has_manufacturing_cost ?y .
   OPTIONAL{?x sol:has_weight ?z}}
```

Table 25 provides the results. Notice that the higher costs and weight belong to the wind deflector (a3_wind_deflector). The weight of the slot bracket and ballast tray pads are not relevant.

Table 25: Manufacturing cost per unit, in US dollars, and weight, in grams.

?part_unit	?unit_cost	?unit_weight
a2_base_rail_right	1.63	1250
a2_base_rail_left	1.63	1250
a2.1_mounting_base_left	1.152	300
a2.1_mounting_base_right	1.152	300
a2.2_bottom_mounting_base_left	0.857	190
a2.2_bottom_mounting_base_right	0.857	190
a3_wind_deflector	5.43	2230
a3.1_slot_bracket	0.1	-
a4_ballast_tray_left	1.045	750
a4_ballast_tray_right	1.045	750
a4.1_ballast_tray_pad_left	0.11	-
a4.1_ballast_tray_pad_right	0.11	-
a4.2_ballast_left	0.2	20000
a4.2_ballast_right	0.2	20000

The second query returns the total manufacturing cost for all the fourteen parts, which is listed as $15.51 dollars, where much of the cost derives from the amount of steel used by the wind deflector. Using a similar query, the total weight of the racking, for one single PV module, is 7.210 kilograms.

Listing 7.24: Query on total manufacturing cost per unit. Design scenario 1.

```
SELECT (SUM(?z) AS ?result)
WHERE
{
   ?y rdf:type sol:PV_racking_assembly ;
      dc:proper-part ?x .
   ?x sol:has_manufacturing_cost ?z .
}
```

By eliminating the wind deflector, the total manufacturing cost is reduced to $9.98. The total weight of the hardware also gets reduced to 4.48 kilograms.

The functional implications produced by the elimination of the wind deflector can be verified in several ways. One of them is by repeating the last query in design scenario 1 (7.22), which returns no results, as expected. given that the query is about the participation of a part that is no longer available in the environment, nothing can be returned.

The same applies to DL queries and defined classes formulated for the inference of aspect systems. For instance, the DL query for the aspect system of the squaring function _fe2.1 _squaring_function, showed in Figure 7.16. Given that no part of the environment has the device-centric function of pre-squaring, the 'squared' state cannot be achieved, and therefore, the resulting aspect system is empty. Moreover, the function _fe2.1_squaring_function is no longer realizable, being classified by the reasoner simply as a _E_function specification.

The conventional solution would be to use measuring tools and methods, involving measuring tape, chalk lines and other auxiliary jigs. This requires to update the specification of the function _fd2_pre−squaring_D_function, which is invoked as precondition of the function _fe2.1_squaring_function, showed below again for convenience.

Listing 7.25: Asserted environment-centric version of squaring function.

```
Individual: sol:_fe2.1_squaring_function
    Facts:
    fr:_EF_in_environment kb:Env1,
    fr:_EF_in_mode_of_deployment sol:_md2.1,
    fr:_EF_nominal_artifact  sol:a1_pv_racking,
    fr:b0  sol:_fd2_pre-squaring_D_function,
    fr:b1  sol:b_squared
```

In the updated version, the input behavior b0 would be the device-centric function _fd2_squaring_by_measure_D_function, in which the nominal artifact is a7_squaring_equipment. Altogether, this equipment has a nominal participation in e2.2b_pre-squaring, qualified by the output behavior b_slow_squaring, as opposed to the output behavior b_integrated_squaring of the wind deflector. The listing shows the assertion for the device-centric function of the chalk line and measuring tape, part of the squaring equipment.

Listing 7.26: Assertion of device-centric function of squaring equipment.

```
Individual: sol:_fd2_squaring_by_measure_D_function
    Facts:
     fr:_DF_artifact  sol:a7_chalk_line,
     fr:_DF_artifact  sol:a7_measuring_tape,
     fr:b0  sol:b_measuring,
     fr:b1  sol:b_normal_squaring
```

Figure 7.19 presents a comparison between the aspect system of the squaring function fulfilled by the wind deflector, according to the design scenario 1, and the aspect system under design scenario 2, where the function is fulfilled by the use of squaring equipment. Notice that the individual a7_squaring_equipment is not mentioned in the specification of the _fd2_squaring_by_measure_D_function above. Yet, since both the chalk line and measuring tape are declared as parts of the equipment, the reasoner also includes the latter.

A more complete overview is provided by the following query expression, which returns not only the aspect system of the function, but also the causal links within the mode of

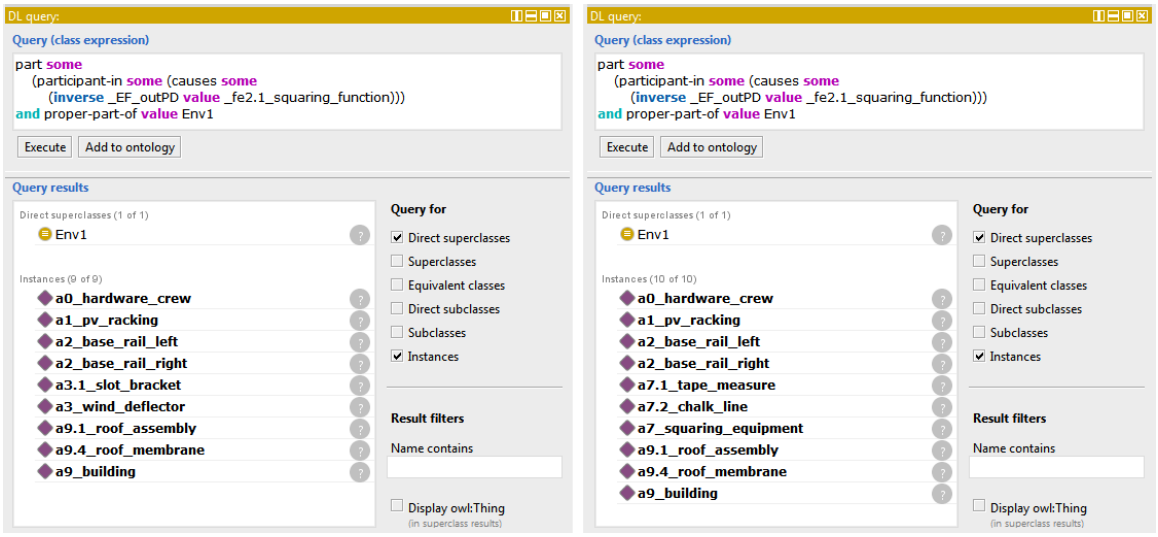

Figure 7.19: Aspect system of squaring function under two scenarios.

deployment _md2.1 and the behavior qualification (i.e. ?by_behavior) for each member of the aspect system, if available. In this case, behavior constraints were also asserted, in terms of average time in seconds for the two squaring methods.

Listing 7.27: Query on qualified aspect system with causal model of squaring function.

```
SELECT DISTINCT (?x AS ?causal_link)(?y AS ?aspect_system)(?b AS ?by_behavior)(?t AS ?secs)
WHERE
{  ?a rdf:type fr:Aspect_system ;
      fr:_AS_aspect_of_function sol:_fe2.1_squaring_function ;
      fr:_AS_co-participant ?y ;
   fr:_AS_in_mode_of_deployment ?m .
   ?m fr:_MD_causal_model_OutPD ?o .
   ?o fr:_c_caused_by ?x .
   ?y dc:proper-part-of kb:Env1 ;
      dc:participant-in ?x .
   OPTIONAL{  ?y fr:_B_has_capability ?b .
              ?b fr:behavior_constraint_on_perdurant ?x .
              ?o fr:_c_caused_by ?x .
              OPTIONAL{  ?y dc:proper-part ?p. ?b sol:has_installation_time ?t}}
   FILTER(?y!=sol:a9_building && ?y!=sol:a9.1_roof_assembly && ?y!=sol:a9.4_roof_membrane)}
ORDER BY desc (?x)
```

A comparison of the results between the two design scenarios is presented in the two tables below. They show the different aspect systems of the squaring function, inferred when using the wind deflector as squaring device first, versus using conventional squaring equipment. They also show the causal model associated with each scenario, along with behavioral qualification whenever it applies. In both query scenarios the building and the rooftop main components have been filtered out of from the results, as indicated by the constraint FILTER in the listing above. This does not mean that they are not part of the aspect systems, but only that they have been excluded for simplification. Thus the results below are consistent with the DL query presented previously in Figure 7.19.

In the first table (26), it can be seen how the quick fastening capability of the slot bracket provides the basis for the integrated squaring capability of the wind deflector. The associated behavioral constraint for squaring performance is the time required to fasten a bracket (30 seconds), and the time to square the racking for one single PV module (45 seconds). It is important to point out once more that if the slot bracket is removed from the wind deflector, then its 'squaring capability' is removed as well. In such a case query returns an empty set for both the aspect system and causal model associated with the squaring function.

These values are not aggregated automatically in the model, but only asserted to the behavioral descriptions associated with the slot bracket and the wind deflector. The goal is to exemplify the use of behaviors to qualify functional participation in terms of performance requirements. SWRL rules or other procedural methods need to be explored in future work, based on a similar method used for the aggregation of total cost and weight presented earlier.

Table 26: Qualified aspect system of squaring function using wind deflector.

?causal_link	**?aspect_system**	**?by_behavior**	**?secs**
e_slotting_1	a3.1_slot_bracket	-	-
e_slotting_1	a3_wind_deflector	-	-
e_slot_fastened	a3.1_slot_bracket	b_quick_fastening	-
e_slot_fastened	a3_wind_deflector	b_quick_fastening	30
e2.2b_pre-squaring	a2_base_rail_right	-	-
e2.2b_pre-squaring	a2_base_rail_left	-	-
e2.2b_pre-squaring	a3_wind_deflector	b_integrated_squaring	45
e2.2b_pre-squaring	a0_hardware_crew	-	-

Table 27: Qualified aspect system of squaring function using squaring equipment.

?causal_link	?aspect_system	?by_behavior	?secs
e2.2b_pre-squaring	a2_base_rail_right	-	-
e2.2b_pre-squaring	a2_base_rail_left	-	-
e2.2b_pre-squaring	a0_hardware_crew	-	-
e2.2b_pre-squaring	a7.1_tape_measure	b_normal_squaring	-
e2.2b_pre-squaring	a7_squaring_equipment	b_normal_squaring	120
e2.2b_pre-squaring	a7.2_chalk_line	b_normal_squaring	-
e5.2_taking_measures	a7.1_tape_measure	-	-
e5.2_taking_measures	a7_squaring_equipment	-	-
e5.3_laying_chalk_lines	a7_squaring_equipment	-	-
e5.3_laying_chalk_lines	a7.2_chalk_line	-	-
e5_pre-squaring_with_measure	a7.1_tape_measure	-	-
e5_pre-squaring_with_measure	a7_squaring_equipment	-	-
e5_pre-squaring_with_measure	a7.2_chalk_line	-	-

It can be seen in table 27 that the aspect system for the same squaring function in the second scenario involves a higher number of parts, as well as a higher number of installation steps and events. In scenario 1, the total number of parts was eight, versus thirteen in scenario 2. More specifically, in scenario 1 no tools or special equipment external to the racking system are necessary, which is not the case for scenario 2 where two tools are required. Notice that the individual a7_squaring_equipment is just an aggregation of the tape measure and the chalk line, used for this example.

The squaring performance of the wind deflector is 45 seconds, while the use of squaring equipment takes 120 seconds (in average). For a large rooftop solar array with more than three hundred PV modules (100kW approximately), the total time of squaring for scenario 1 would be 4.5 hours, versus 12 hours of scenario 2 using conventional squaring methods.

Certainly, there are other indicators that might be needed to fully qualify the participation of different squaring devices, besides average squaring time per module. For instance, the level of training required to perform the task is a relevant factor because it impacts overall installation costs. Concurrency in the task might also be relevant, in the sense that integrated squaring allows modules to be attached almost at the same time. Conversely, in conventional squaring, entire areas of the array need to be squared before any module can be attached, adding to the total time of installation, and consequently, impacting cost.

These indicators can be added to the knowledge base by means of extra assertions

made the each behavioral descriptor. As example, the behavior b_integrated_squaring could be further enhanced according to the listing below. This mechanism could support additional query scenarios, including those required for the selection and retrieval of parts from component libraries.

Listing 7.28: Enhanced behavioral description for integrated squaring capability.

```
1 Individual: sol:b_integrated_squaring
2   Types:
3     fr:Behavior
4   Facts:
5     sol: has_installation_time  45,
6     sol: has_required_training_level  sol:low,
7     sol:has_concurrency  "true"^^xsd:boolean
```

7.3 Case study 3: Ballasted PV racking system - Part B

The third and final case study is an extension of the previous one, addressing additional functional aspects of the PV racking system, including interactions with the functionality of the building roof. It also addresses the functional implications of design modifications in different phases of the product life-cycle. These modifications involve for the most part the addition and removal of features and components, providing a series of scenarios that demonstrate the capabilities of the proposed framework to support incremental elucidation of aspect systems.

In particular, the design scenarios developed in this case study allow to illustrate how the functional modeling approach adopted, and the inference capabilities embedded in the model, allow to identify new sets of unanticipated 'lurking' requirements, that emerge as result of last minute design decisions.

The ability to describe and infer distal functional interactions were also explore this case study, related with the multiple functional roles of the wind deflector design. These distal functional interactions and implications are captured in two main trade-off design scenarios developed for this case study:

- Functional aspects associated with electrical grounding of metal parts.
- Structural aspects associated with lateral loads (e.g. seismic loads).

7.3.1 Identified task: design scenarios

The first aspect is result of the requirements imposed by the standard UL1703, which demands that all electrically conductive parts of significant size need to be electrically bonded for grounding. This requirement applies to the wind deflector, which needs to be fastened to the mounting supports of the base rails in such a way that an electrical ground path is maintained. In concrete terms this means that the electrical resistance between parts needs to be less than 200mΩ. Typically, this is achieved by means of grounding lugs, special rivets or star washers tightened at certain torque using bolts or screws.

The use of a screw bracket, discussed in the previous case study, would allow the implementation of the latter grounding solution, which is not true in the case of slot brackets. However, a screw bracket implies more parts to be procured and handled on-site, requiring tools and additional installation steps not needed in the case of the slot bracket. On the other hand, a slot bracket behaves as a pin connection, whereas the screw bracket can be a moment-resistant connection, which is more suitable for lateral bracing of the solar array.

Finally, in the situation where the wind deflector is not used at all, uplift loads produced by wind need to be mitigated with additional ballast. Depending on the structural system of the roof, the extra weight can cause excessive deflection, which might lead to rainwater accumulation in depressed areas of the roof surface, eventually causing water leaks inside the building.

Hence, seemingly minor design decisions such as those involving bracket types, or wind deflectors, can produce unintended consequences not only across different building subsystems, but also across levels of abstraction. The goal of this case study is to illustrate how the proposed functional modeling framework can capture these functional inter-dependencies by inference of aspect systems involved in each scenario. The following list summarizes these scenarios and the functional trade-offs involved in each of them:

1. Trade-off 1: Electrical bonding and installation performance:
    - Slot bracket
    - Screw bracket (with star washer)
2. Trade-off 2: Bracing and installation performance:
    - Wind deflector with slot bracket (pin connection)
    - Wind deflector with screw bracket (moment-resistant connection)
    - No wind deflector
3. Trade-off 3: Gravity loads and building envelope integrity:
    - Wind deflector and low ballast weight
    - No wind deflector and high ballast weight

The next subsection presents the formalization of the different problem scenarios by means of encoded assertions.

7.3.2 Problem formalization and encoding

The functional model for this case study is an extension of the previous one, developed for the case study 2 to the one adopted in the previous case study. As such, it consists of a series of assertions about classes, properties and rules, as well as assertions of facts about individuals at the instance level, both endurants and perdurants.

7.3.2.1 Main assertions

The first listing below contains assertions about the electrical bonding function of star washers, which are used as part of a screw set, which in turn is a part of the screw bracket. This is followed by the specification of bonding capability behavior in listing 7.30, which qualifies the participation of the star washer in the bonding process. In particular, this group of assertions specifies a behavioral constraint for that participation, in terms of an electrical resistivity value of 20mΩ (20 Milliohms). In the same listing, the behavior b_tighted_by_torque is asserted, which is the precondition specified for the bonding functionality of the star

washer. Finally, at the bottom, structural constraints are added, in terms of direct connection of the star washer with mounting supports of the base rails.

Listing 7.29: Assertions about screw set, with a star washer for electrical bonding as part.

```
1  Individual: sol: _fd5_electrical_bonding
2      Facts:
3          fr:_DF_artifact  sol:a8_star_washer,
4          fr:b0 sol:b_tighted_by_torque,
5          fr:b1 sol:b_bonding_by_torque
6  Individual: sol:a8_screw_set
7      Facts:
8          dc:proper-part sol:a8_screw,
9          dc:proper-part sol:a8_star_washer,
10         dc:proper-part-of: sol:a3.1_screw_bracket,
11         dc:participant-in sol:e_screwing_3,
12         fr:nominal_participant_in  sol:e_screwing_2
13 Individual: sol:a8_star_washer
14     Facts:
15         fr:s_directly_connected_to  sol:a2.1_mounting_base_left,
16         fr:s_directly_connected_to  sol:a2.1_mounting_base_right
```

Listing 7.30: Assertions about bonding capability of star washer with behavioral constraint.

```
Individual: sol:b_bonding_by_torque
    Types:
        sol:Bonding_capability
    Facts:
        fr:behavior_constraint_on_perdurant sol:e3.1 _electrical_bonding ,
        sol: has_electrical_resistivity 20 // In Milliohms
Individual: sol:b_tighted_by_torque
    Facts:
        fr:behavior_constraint_on_perdurant sol:e_screwed_fastened
Individual: sol:e_screwed_fastened,
    Facts:
        fr: _c_causes sol: e_piercing
Individual: sol: e_piercing
    Facts:
        fr: _c_causes sol:e3.1 _electrical_bonding
Individual: sol:e3.1 _electrical_bonding
    Facts:
        fr: _c_causes sol:e3.1 _electrical_continuity
```

For the screwing processes and events, two types of screw drivers have been asserted. The first is a manual screw driver (screw_driver_1), while the second is a battery-powered with torque control screw_driver_2). This was considered necessary to assess the often tacit implications of using star washers for electrical bonding in the aspect system associated with the installation performance. Moreover, these assertions allow to exemplify how distal functional interactions, for which no obvious structural relations exist, can be represented according to the formalism proposed.

In both cases, no complete specification of device functions was made, but only assertions about qualified participation which describe output behaviors in terms of relevant behavioral capabilities. Both types of screw drivers, along with the perdurants (i.e. EPD) in which they participate are described as follows:

Listing 7.31: Behavioral qualification of battery-powered screw driver.

```
Individual: sol:a8_screw_driver_1
    Facts:
        fr:_B_has_capability  sol:b_manual_screwing
Individual: sol:b_manual_screwing
    Facts:
        sol:behavior_constraint_on_perdurant e_screwing_3
        sol:has_torque_control  "false"^^xsd:boolean,
        sol: has_installation_speed  "slow"^^xsd:string
Individual: sol:a8_screw_driver_2
    Facts:
        fr:_B_has_capability  sol:b_battery_powered_screwing,
        dc:proper−part sol:battery
Individual: sol:b_battery_powered_screwing
    Facts:
        sol:behavior_constraint_on_perdurant e_power_screwing,
        sol:has_torque_control  "true"^^xsd:boolean,
        sol:has_torque 30,                          // In Newton−meters (N.m)
        sol:has_voltage_requirement 20,             // In Volts
        sol:has_speed 1450,                         // In RPM
        sol:has_charging_time 60,                   // In Minutes
        sol: has_installation_speed  "fast"^^xsd:string
Individual: sol:e_power_screwing
    Facts:
        fr:_d_causes  sol:e_screwing_3
```

The remaining of the assertions required to model the trade-off scenarios of this case study are described in the appendix X. In general, the remaining assertions follow the same vocabulary and modeling criteria discussed so far, addressing the representation of structural capabilities of the two bracket types considered for the wind deflector (i.e. pin and moment-resistant connections), bracing and buckling perdurants, as well as assertions about uplift, deflection, water accumulation, drag, tear and leakage of the roof.

7.3.3 Design trade-off scenario 1

The general pattern described in this scenario addresses the functional implications of design changes produced by the addition of new elements to the system. In such situations it is common for added elements to carry over a new sets of functional requirements usually not anticipated by stakeholders involved. This contrasts with the pattern addressed in the previous case study, where functional implications were produced by the removal of elements with multiple functions. Another issue addressed in this scenario is the inference of distal functional interactions, which cannot be easily captured by structural relations and constraints. Instead, they require inference of common causal links within a given mode of deployment. This problem will be addressed in the last part of this scenario, which deals with the elucidation of hidden components of aspect system caused by the addition of new design elements. The assertions about manual and automatic screw drivers presented in the last listings will be used to that purpose.

The specific design modification triggering this trade-off scenario is the addition of screw brackets to the wind deflector, to complement the initial attachment solution provided by the slot brackets. This is considered necessary to improve two extra performance requirements. The first is to provide a more reliable bonding solution for electrical grounding. Since the wind deflector is made of metal, it is required to be grounded. While the slot bracket does provide some level of electrical continuity to the base rails, necessary for grounding, the contact is not reliable enough. A screw bracket in turn can provide the necessary degree of bonding between wind deflectors and base rails by relying on star washers that can maintain the electrical continuity over time, and under a wide range of environmental conditions.

The second advantage of screw brackets, to be addressed in the next trade-off scenario, deals with the moment-resistant capabilities of this connection. This is necessary to provide bracing for the solar array under lateral loading conditions.

The electrical grounding requirements are declared under the environment-centric function _fe5.2_electrical_grounding, which takes the device-centric function _fd5_electrical_bonding as input behavior (b0). A SPARQL query similar to the one used previously for the squaring function was used to elucidate the aspect system of the electrical safety function. In this case

however, it is necessary to point out an important difference in the SPARQL expression. In this query, not only the function is different, but also the environment of the function. Indeed, the squaring function relates to the environment of the installation process, involving installation crew, equipment and tools among other endurants specific to that task. Such installation specific environment was denoted previously by the individual Env1 (in line 9 of listing 7.27). Hence, all the query results about the squaring function belong to that environment. On the other hand, the queries about the electrical safety function belong to the operational environment of the solar array, which takes place only after installation. This is denoted by the individual Env2, which was used to formulate the SPARQL query for the electrical bonding function. In this way, different environment conditions can be described in the proposed framework, each corresponding to a specific life-cycle stages of an artifact or system.

Table 28: Qualified aspect system of grounding function. Resistivity in Milliohms.

?causal_link	**?aspect_system**	**?by_behavior**	**?resistivity**
e_screwing_2	a3_wind_deflector	-	-
e_screwing_2	a3.1_screw_bracket	-	-
e_screwing_2	a3.1_screw_set	-	-
e_screwing_3	a3.1_screw_set	-	-
e_screwing_3	a3.1_screw_bracket	-	-
e_screwing_3	a3_wind_deflector	-	-
e_screwed_fastened	a3_wind_deflector	b_screwing	-
e_screwed_fastened	a3.1_screw_bracket	b_screwing	-
e_electrical_continuity	a2_base_rail_right	-	-
e_electrical_continuity	a2_base_rail_left	-	-
e_electrical_bonding	a3_wind_deflector	b_bonding_by_torque	20
e_electrical_bonding	a3.1_slot_bracket	-	-
e_electrical_bonding	a3.1_screw_bracket	b_bonding_by_torque	20
e_electrical_bonding	a3.1_screw_set	b_bonding_by_torque	20
e_electrical_bonding	a3.1_star_washer	b_bonding_by_torque	20

As it can be seen, both slot bracket and the screw bracket are part of the aspect system of electrical grounding. In particular, the latter includes all its internal components, such as the star washer that fulfills the device-centric bonding function, and the screw itself. As mentioned before, the slot brackets have a participation in electrical bonding that is not nominal nor qualified. This is not the case for the screw bracket, which inherits the bonding capability of the star washer, including its electrical resistivity of 20mΩ.

This differentiation allows the formulation of more specialized queries based on various behavioral capabilities and constraints. For instance, given a number of parts with participation in electrical bonding, only those with values above or below certain threshold can be selected. Other queries might involve a combination of different criteria. For example, the first query below selects all parts with electrical bonding capabilities. In such a case, not only the screw bracket and its star washer will be returned, but also the wind deflector and the entire racking system as a whole. This is because behavioral capabilities are accumulated into higher levels of the structural composition. Thus, if one minor part has a capability, the entire system has the capability. This is not to say that the entire system performs the capability at the same level, but only that there is at least one component in the assembly that does, at least theoretically. The final evaluation of performance is a separated task, to be done by analysis or simulation. This however is considered an first useful approach to harvest parts and properties with cross-cutting functional implications.

Listing 7.32: SPARQL query on parts with bonding capability only.

```
1 SELECT (?x AS ?aspect)
2 WHERE
3 {  ?a rdf:type fr:Aspect_system ;
4    fr:_AS_aspect_of_function sol:_fe5.2_electrical_grounding .
5    ?x fr:member_of_aspect_system ?a ;
6    fr:_B_has_capability ?b .
7    ?b a sol:Bonding_capability }
```

As it would be expected, the slot bracket is not returned, since its participation in bonding is not qualified. The second query below in turn selects parts that have both a fast attachment capability and also participation in electrical bonding. In this case, only the slot bracket is returned.

Listing 7.33: SPARQL query on quick fastening parts that participate in bonding.

```
SELECT (?x AS ?aspect)
WHERE
{  ?a rdf:type fr:Aspect_system ;
   fr:_AS_aspect_of_function sol:_fe5.2_electrical_grounding .
   ?x fr:member_of_aspect_system ?a ;
   dc:participant-in sol:e_electrical_bonding ;
   fr:_B_has_capability sol:b_quick_fastening }
```

Figure 7.3.3 presents these queries as done in Protégé using the Snap SPARQL plug-in. The list of results is presented below each of these two queries, in the same corresponding window.

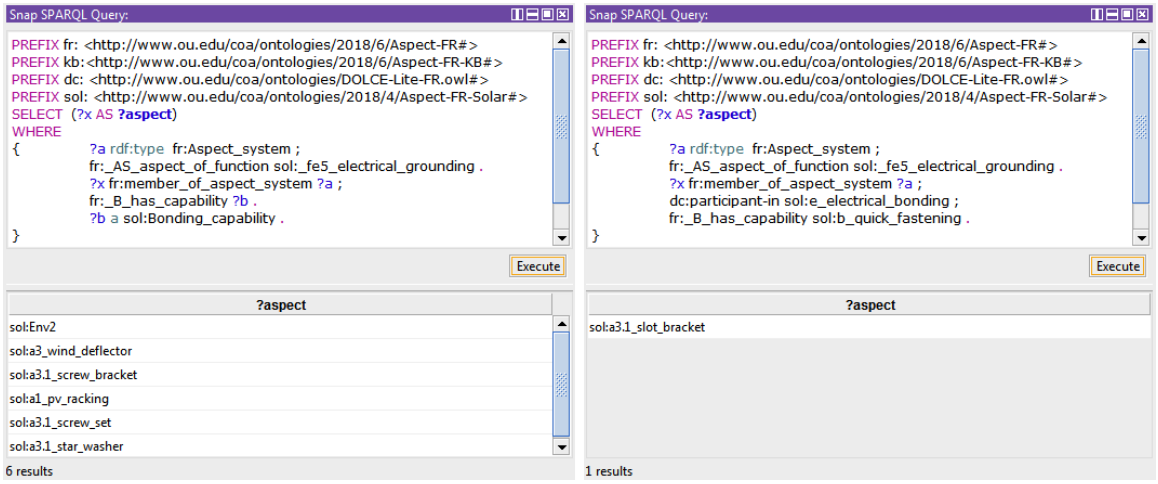

Figure 7.20: Query on parts with asserted bonding capability (left). Query on parts with quick fastening capability and participation in bonding (right).

One of the goals of the functional modeling framework proposed in this research was the ability to support different functional views at various levels of abstraction. There are a number of ways in which this can be done. The first is by specifying the target output perdurant within a chain of causal links. Thus, functions can be specified and aspect systems inferred depending on the position of the output perdurant in the chain. To make an analogy with a linked list data structure, the longer the list and the more far away the head of the list is from the bottom node, the larger the number of potential participants

that make up the aspect system of the function. The functional modeling schema presented here allows, to specify that position, such the elucidation of aspect system with different levels of granularity are possible. This approach was initially demonstrated in the first case study about the electronic device.

A second approach involves switching between device-centric and environment-centric functional views. In this approach, the device itself can be considered as the environment. While this was done implicitly in the first case study, the difference was formalized in the second case-study. By switching perspectives, different levels of detail can be obtained from both mode of deployments and aspect systems.

The third approach, which complements the other two, involves switching functional perspectives and environments altogether. The electrical grounding function represents a perspective that belongs to the operational environment of the solar array. Therefore, aspects systems inferred for this function contain only members that are normally part of that environment. However, the addition of screw brackets for bonding has functional implications in other stages of the racking life-cycle, either before operations of the solar system, or after (e.g. maintenance).

To assess these implications, a query similar to a previous one made to infer the aspect system of the squaring function at the end of the second case study was used. In this case however, a different representation was adopted to capture the aspect system of the grounding function. Instead of relying of property chains as done previously, the aspect system was captured by a defined class called Aspect_system_grounding_function using with the following equivalence axiom:

Listing 7.34: OWL defined class for qualified aspect system of grounding function.

```
1 dc:part some
2   (dc:participant-in some
3     (fr:_c_causes some
4       ( inverse (fr:_DF_outPD) value sol:_fe5.2 _electrical_grounding ))))
5  and dc:proper-part-of value kb:Env1
```

Based on this defined class, the new SPARQL expression is slightly different than the one used previously in case study 2. First, the squaring function _fe2.1Squaring_function is used again (in line 3 of the listing below) to made to 'cap' the reach of the causal model. Then, the defined class for the aspect system of the grounding function is used in line 7.

Listing 7.35: SPARQL query for aspect system of squaring function

```
1  SELECT DISTINCT (?x AS ?causal_link)(?y AS ?aspect_part)(?b AS ?by_behavior)(?t AS ?speed)
2  WHERE
3  {  ?f owl:sameAs sol:_fe2.1_squaring_function ; # Change for _fe5_electrical_installation
4        fr:_DF_outPD ?o .
5     ?x fr:_c_causes ?o;
6        dc:participant ?y .
7     ?y a sol:Query_aspect_system_grounding .
8     OPTIONAL{  ?y fr:_B_has_capability ?b .
9              ?b fr:behavior_constraint_on_perdurant ?x .
10             OPTIONAL{  ?y dc:part ?p .
11                     ?b sol:has_installation_speed ?t }}
12 FILTER(?y != sol:a1_pv_racking && ?y != sol:a8_installation_tools)}
```

Since both slot brackets and screw brackets are used now as part of the wind deflector (the latter was added to the design), and the squaring equipment is no longer needed, the returned aspect system for the squaring system remains the same. This is because the process of fastening the screw brackets occurs at the end of the electrical installation process, given that one of functions of this component is to provide bonding for grounding the wind deflector. By changing the specification of the 'cap' in line 3 of the query expression above by the function _fe5_electrical_installation , the resulting aspect system is showed in the Table 29 below:

The table shows a different scope of the installation process, that happens after squaring and attachment of PV modules. In particular, the decision was made to screw the wind deflector to the mounting rails after the electrical wires were connected and bundled together, a process called 'wire management'. This is result of a process planning decision. In a different scenario, the screwing could be made immediately after squaring.

Table 29: Qualified aspect system of electrical installation function.

?causal_link	?aspect_part	?by_behavior	?speed
e_charge	a8_power_outlet	b_supply_AC_power	-
e_charge	a8_screw_driver_2	b_portable_power_supply	-
e_charge	a8_battery	b_portable_power_supply	-
e3.1_wiring	a0_electrical_crew	-	-
e3.2_wire_management	a5_zip_ties	-	-
e3.2_wire_management	a3_wind_deflector	b_integrated_wire_support	-
e_screwing_2	a3.1_screw_set	-	-
e_screwing_2	a3_wind_deflector	-	-
e_screwing_2	a3.1_screw_bracket	-	-
e_screwing_3	a3.1_screw_set	-	-
e_screwing_3	a3_wind_deflector	-	-
e_screwing_3	a8_screw_driver_1	b_manual_screwing	“slow”
e_screwing_3	a3.1_screw_bracket	-	-
e_screwed_fastened	a3_wind_deflector	b_screwing	-
e_screwed_fastened	a3.1_screw_bracket	b_screwing	-
e_power_screwing	a8_screw_driver_2	b_battery_powered	“fast”
e_electrical_bonding	a3.1_slot_bracket	-	-
e_electrical_bonding	a3.1_screw_set	b_bonding_by_torque	-
e_electrical_bonding	a3.1_star_washer	b_bonding_by_torque	-
e_electrical_bonding	a3_wind_deflector	b_bonding_by_torque	-
e_electrical_bonding	a3.1_screw_bracket	b_bonding_by_torque	-
e_electrical_continuity	a2_base_rail_right	-	-
e_electrical_continuity	a2_base_rail_left	-	-

In any case, the results show that the decision of adding screw brackets to the wind deflector lead not only to more installation steps, but also to a different aspect system involving more parts. Now, the two types of screw driver asserted before are included, as denoted by the entries a8_screw+_driver_1 and a8_screw_driver_2. Moreover, for the automatic screw driver, which enables a faster screwing process, the battery is also included. This behavioral characterization of performance allows to select which type of screw driver to use in a given installation. Notice that the power outlet a8_power_outlet is also listed, related to the screw driver and battery by means of the common causal link e_charge.

Indeed, the choice for a faster, automatic screw driver implies a more complex aspect system for the whole installation function than just batteries and possibly different screw bits. The inclusion of the power outlet in the results indicates the possibility of other hidden parts of the aspect system that need to be elucidated. This can be done by querying the causes and co-participants of the perdurant e_power_screwing to expose the pre-requisites needed to make the automatic screw driver work as intended.

To do so, the query cannot be limited to devices that are part of the environment, since its complete composition might be unknown. The following SPARQL expression presents the query used, which also includes the different behavioral constraints associated with each hidden co-participant. These in turn allow to assess other pre-requisite and functional requirements that will be brought into the process by the addition of the automatic screw driver.

Listing 7.36: Query for hidden parts of aspect system for electrical installation function.

```
SELECT DISTINCT
(?y AS ?hidden_part)(?b AS ?by_behavior) (?c AS ?_constraint)(?v AS ?_value)(?u AS ?unit)
WHERE
{  ?a owl:sameAs sol:e_power_screwing;
      fr:_c_caused_by ?o.
      ?o dc:participant ?y .
   OPTIONAL { ?y fr:_B_has_capability ?b .
            OPTIONAL { ?y dc:part ?p .
                    ?c a owl:DatatypeProperty ;
                    rdfs:label ?u ;
                    rdfs:subPropertyOf sol:has_behavioral_property .
                    ?b ?c ?v . }}
FILTER (?y != sol:a1_pv_racking && ?y != kb:Env1 && ?y != sol:a8_installation_tools)}
```

While this query could have been integrated with the previous one, such integration would be difficult to express, and computationally more expensive. In particular, such integrated query would require to specify the hidden parts of the aspect system using negation. This means a clause indicating individuals that participate in e_power_screwing but that are *not* part of the environment. Given that OWL operates under an Open World Assumption (OWA), there is no direct or simple way to express that condition. Therefore, this last query avoids mentioning the condition altogether. While this option is useful for the purposes of modeling this particular scenario, in other situations it could lead to a long list of irrelevant results.

In practice, the hidden parts elucidated by the query are pre-requisites for the functioning of the automatic screw driver. Furthermore, each hidden part imposes a new set of behavioral constraints and functional requirements to be satisfied during installation. Among these, there is the need for a 220V power outlet within the reach of power extension cords (typically under 100' or 30 meters long). Often these types of constraints are not taken in consideration in the design and planning process of a solar installation, or construction activity for that matter. Table 30 presents the results:

Table 30: Hidden members of aspect system for electrical installation function.

?hidden_part	**?by_behavior**	**?_constraint**	**?_value**	**?unit**
a8_extension_cord	b_conductivity	has_extension	30	m
a8_extension_cord	b_conductivity	has_conductor_gauge	12/3	Wire gauge
a8_extension_cord	b_conducts_electricity	has_max_amps	15	Amps
a8_power_outlet	b_supply_AC_power	has_power_supply	110	V
a8_charger	b_charging_capability	has_voltage_required	220	V
a8_charger	b_charging_capability	has_charging_time	60	Minutes
a8_battery	b_portable_power_supply	has_supply_cycles	500	Cycles
a8_screw_driver_2	b_battery_powered	has_voltage_required	20	V
a8_screw_driver_2	b_battery_powered	has_torque	30	N.m
a8_screw_driver_2	b_battery_powered	has_installation_speed	"fast"	Install speed
a8_screw_driver_2	b_battery_powered	has_torque_control	"yes"	Yes / No
a8_screw_driver_2	b_portable_power_supply	has_supply_cycles	500	Cycles

All returned hidden parts are functionally related to the automatic screw driver (a8_screw_driver_2) by the common causal link e_charge. Evidently, they would all be included in the aspect system of the electrical installation function if they had been made part of the environment explicitly in first place. For instance, the battery charger and the extension cord could be inferred as part of the environment automatically if they were asserted or inferred as structurally connected to the power outlet. This inference could be based on a an equivalent class or SWRL rule, grounded on the relations fr : s_connected_to, fr : s_directly_connected_to .

In any case, some final observations can be made from the previous table. First, even though a power supply might be available near the rooftop, it could be beyond the reach of a single extension cord. This would require additional extension cords to be brought to the site. The second observation is that the battery charger requires a 220V power source, while the outlet provided is 110V. Finally, the number of screwing cycles of the screw driver

is limited to 500. This is a behavioral constraint from the battery, which is asserted as part of the screw driver. In this functional modeling framework, behavioral capabilities and constraints of parts are assigned to the top level of the device structural hierarchy. Thus, given a more capable battery in terms of cycles, the screw driver would be able to deliver more screwing events between charges. On the other hand, a more powerful charger would be able to fully charge a battery in less than sixty minutes, etc.

7.3.4 Design trade-off scenario 2

The previous design scenario deals with the functional implications of design changes that could be considered as negative, such as the extra logistic cost added by the use of power tools during installation. The current scenario in turn deals with positive implications of the same design changes. In particular, the addition of screw brackets affords their use not only as electrical bonding devices, but also as moment-resistant connections. This enables a more rigid attachment between wind deflectors and mounting bases, providing bracing under lateral loads caused by wind and seismic movements. The main advantage of adding bracing is that the resulting stresses are absorbed by the wind deflectors instead of the PV modules. This is necessary to avoid damage to the module frames, and more importantly, to avoid micro-cracks in the solar cells, which might compromise the energy performance of the entire system.

In order to represent the phenomena and the functional aspects involved, a series of refinements have to be made to the previous model, along with new assertions regarding composition and behavioral capabilities of the screw brackets and wind deflectors. The most important difference however is the use of cardinality restrictions in certain structural and causal relations, imposed over the corresponding OWL object properties. In this case, it is necessary to specify that a bracket requires at least two fasteners to become a moment-resistant connection. For instance, the screw bracket used in the previous scenario requires at least two screws. Having just one screw is enough to provide electrical bonding, but not to provide moment resistance capabilities. In such situation the bracket defaults into a pin connection, behaving like a hinge under lateral loads.

The wind deflector in turn needs to be connected to two different but adjacent mounting supports, with at least one of these connections being moment-resistant. Hence, if both screw brackets have only one screw each, then the wind deflector loses its ability to fulfill a bracing function.

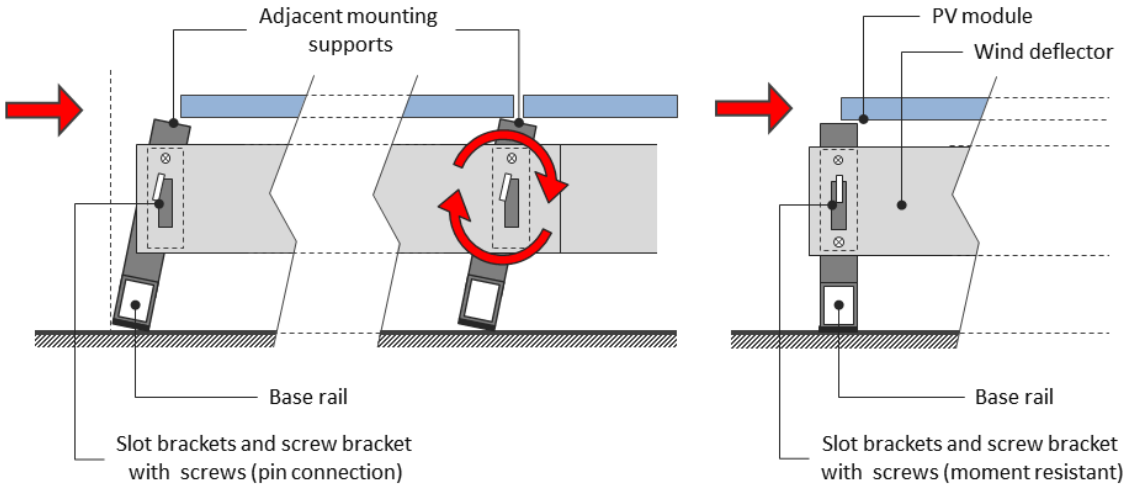

Figure 7.21: Side view of PV racking. Screw brackets with only one screw do not allow wind deflector to brace the mounting supports (left side). Bracing capability of the wind deflector provided by adding a second screw to the bracket (right side).

To list below summarizes the rationale followed in the development of the functional model for this scenario. This provides the criteria for the queries used for model validation:

1. Each wind deflector has a pair of slot brackets for integrated squaring.
2. Each wind deflector has a pair of screw brackets for electrical bonding.
3. Slot brackets are pin connections.
4. Screw brackets with only one screw are pin connections.
5. Screw brackets with more than one screw are moment-resistant connections.
6. At least one moment-resistant connection enables wind deflector as bracing device.
7. A braced racking system contributes to maintain the form of the solar array.
8. A ballasted racking system contributes to maintain the position of the solar array.

9. Form and position contributes to the overall structural integrity of the solar array.

The additional assertions required for modeling this design trade-off scenario involve specification of two new device capabilities. The first one is about the moment-resistant capability of connections in general, and the second about bracing capabilities. The former was defined not as an explicit device-centric function, but just as a behavioral capability, following the approach used in the specification of the manual and automatic screw drivers in the previous scenario.

In this case however, the brackets are not asserted, but inferred as having moment-resistant capabilities based on the cardinality restriction of having at least one fastener (e.g. screw, bolt or rivet). Any bracket that fulfills that restriction, or is welded, is classified as a sub-class of connections that are moment-resistant. Direct assignment of moment-resistant capability to the bracket is showed in the listing below, which specified the cardinality constraint as equivalent class axiom.

Listing 7.37: Axiom for inference of moment-resistant connection based on cardinality.

```
1 Class: sol:Moment_resistant_connection
2     EquivalentTo:
3         (fr:has_component some sol:Welded_connection)
4         or (fr:has_component min 2 sol:Fastener)
5     SubClassOf:
6         sol:Pin_connection,
7         fr:_B_has_capability value sol:b_moment_resistence_capability
```

As for the bracing capability, this was specified as complete functional specification, in such a way that any device fulfilling the cardinal restrictions over bracing would be allocated the device-centric bracing function. The listing below shows the equivalence axiom with the conditions for class membership, and the subclass axiom specifies the implication, in terms of the functionality allocated to classified members. Notice that the cardinal restriction of min 2 does not refer directly to the Mounting_support type, but instead refers to the nested sub-condition that such mounting supports need to adjacent to each other.

Listing 7.38: Axiom for inference of device with bracing capability based on cardinality.

```
1 Class: sol:Bracing_device
2     EquivalentTo:
3         (fr:has_component min 1 sol:Moment_resistant_connection)
4          and (fr:s_connected_to min 2 (fr:s_adjacent_to some
5                 sol:Mounting_base_top))
6     SubClassOf:
7          inverse (fr:_DF_artifact) value sol:_fd4.2_bracing
```

In other words, this definition means that any technical artifact connected to at least two adjacent mounting bases, and having at least one moment resistant connection, has a participation in bracing. Such participation is a qualified, nominal participation, as it is specified by the assertion that any member of the class becomes the _DF_artifact of the function _fd4.2_bracing, showed in line seven of the listing. Certainly, it could be argued that other conditions are required to fully characterize the class of bracing devices. However, the intent here is to demonstrate the working principles of the modeling framework, and not the correctness or validity of this assertions from a disciplinary perspective.

To support this approach, the assertion of the bracing function _fd4.2_bracing does not include any prior specification of nominal artifact, as done in the previous scenario. instead, nominal participants are inferred by the reasoner according to the equivalent class formulated above. Therefore, the function specification can be limited to the assertion of input and output behaviors, as follows:

Listing 7.39: Assertion of device-centric bracing function.

```
1 Individual: sol:_fd4.2_bracing
2     Facts:
3       fr:b0   sol:b_loading_seismic,
4       fr:b0   sol:b_loading_wind,
5       fr:b1   sol:b_bracing_capability
```

The assertion means that any allocated nominal artifact will have a qualified participation in wind and seismic loading, as input perdurants, and a qualified participation in bracing. That is, the artifact is expected to take in wind and seismic loads with certain behavioral constraints as input (e.g. maximum wind speed of 80 mph), and deliver or maintain certain bracing performance as output. The nature and value of these constraints have not been addressed in this case studio, and future work will be required to provide the necessary type of domain semantics.

Bracing capabilities, along with ballasting and friction provided by ballast tray pads to resist wind drag, allow to maintain the form and position of the entire rooftop solar array, which is required to maintain the overall structural integrity of the system. The specification of device-centric functions for ballasting and drag resistance follow a similar approach used for the bracing function, and therefore do not need to be discussed here. The full model specification is given in Appendix E.

The need to maintain form and position of the array, as key pre-requisites to the entire structural integrity of the array, exemplifies how functions are built on top of lower level functions, across multiple levels of abstraction, from different environment-centric perspectives, down to device-centric viewpoints. Assertions about these three high-level functions and how they relate to each other is given according to the next listing. The first function asserted is _fe7.1_maintain_position, followed by the function _fe7.1_maintain_form, and the top function _fe7_maintain_structural_integrity .

Listing 7.40: Assertions for the environment-centric functions for structural integrity.

```
1 Individual: sol:_fe7.1_maintain_position
2     Types:
3         fr:_E_function
4     Facts:
5         fr:_EF_in_environment              kb:Env1,
6         fr:_EF_in_mode_of_deployment       sol:_md7,
7         fr:_EF_nominal_artifact            sol:a1_pv_racking,
8         fr:b0    sol:_fd4.1  _resist_drag_by_friction  ,
```

fr : b0 sol : _fd4.1 _resist_uplift_by_ballast ,

fr : b0 sol : _fd4.1 _resist_uplift_by_deflection ,

fr : b1 sol : b_maintain_position_capability

Individual: sol: _fe7.2_maintain_form

Types:

fr : _E_function

Facts:

fr : _EF_in_environment KB:Env1,

fr : _EF_in_mode_of_deployment sol : _md7,

fr : _EF_nominal_artifact sol : a1_pv_racking,

fr : b0 sol : _fd4.2_bracing,

fr : b1 sol : b_maintain_form_capability

Individual: sol: _fe7_maintain_structural_integrity

Types:

fr : _E_function

Facts:

fr : _EF_in_mode_of_deployment sol : _md7,

fr : _EF_nominal_artifact sol : a1_pv_racking,

fr : _EF_nominal_artifact sol : a9.1_roof_assembly,

fr : b0 sol : b_maintain_form_capability,

fr : b0 sol : b_maintain_position_capability ,

fr : b1 sol : b_maintain_structural_integrity_capability

The downstream functional dependency of the top function e7_structural_integrity is controlled by a cardinal restriction over its direct causal pre-conditions, namely, e7_maintain_position and e7_maintain_form. The restriction is specified as part of the mode of deployment of the structural integrity function, which is defined in terms of the following equivalent class axiom.

Listing 7.41: Cardinal restriction on causal preconditions for state of structural integrity.

```
Class:
    fr : Maintaining_structural_integrity
     EquivalentTo:
          fr : caused_directly_by  min 2
          sol : Maintaining_form_and_position
     SubClassOf:
          fr : CauseMD, Aspecto−FR:causes value
          sol :  e7_structural_integrity
```

This axiom formalizes the output perdurant state of e7_structural_integrity as the results of two causal preconditions, which in turn are dependent on lower level functional preconditions. Thus, by removing one screw from each bracket of a wind deflector, this device loses its bracing capabilities, and by consequence the maintenance of form can no longer be guaranteed, but only the maintenance of position, thanks to the ballast trays. This however is enough to compromise the structural integrity of the solar array, with potential negative implications to the rooftop of the building.

To demonstrate the ability of the proposed framework to support the elucidation of aspect systems associated to each function, a series of queries have been developed following the rationale presented earlier. In particular, the queries follow a sequence of design modifications, starting with a complete state of the design in which all device functions are satisfiable, and all environment functions are realizable. Then, parts will be gradually removed from the design in order to assess the functional implications and the changes on the corresponding aspect systems.

Table 31 presents the entire set of satisfiable and realizable functions of the initial state of the PV racking design. This state is complete, in the sense it contains all parts defined. Corresponding queries in SPARQL return 10 satisfiable device functions, 11 satisfiable environment functions, and 10 realizable functions for this complete state of design.

Table 31: Full set of satisfiable and realizable functions in flat roof PV design.

?satisfiable_D_function	**?satisfiable_E_function**	**?realizable_E_function**
_fd1_flat_packing_D_function	_fe0_flat_packing	_fe0_flat_packing
_fd2.1_screw_fastening_D_function	_fe1_whole_installation	_fe1_whole_installation
_fd2.1_slot_fastening_D_function	_fe2.1_squaring_function	_fe2.1_squaring_function
_fd2_pre-squaring_D_function	_fe2.2_pv_module_attachment	_fe2.2_pv_module_attachment
_fd2_squaring_by_measure_D_function	_fe2_hardware_installation	_fe2_hardware_installation
_fd3_wire_management_D_function	_fe5.1_wire_management	_fe5.1_wire_management
_fd4.1_resist_uplift_by_ballast	_fe5.2_electrical_grounding	_fe5.2_electrical_grounding
_fd4.1_resist_uplift_by_deflection	_fe5_electrical_installation	_fe7.1_maintain_position
_fd4.2_bracing	_fe7.1_maintain_position	_fe7.2_maintain_form
_fd5_electrical_bonding	_fe7.2_maintain_form	_fe7_maintain_structural_integrity
	_fe7_maintain_structural_integrity	
total: 10	**total: 11**	**total: 10**

The design modifications made to assess cross-cutting functional implications at various levels of abstraction consider the following four steps:

1. One screw is removed on each of the two screw brackets of the wind deflector.
2. Screw brackets are removed from wind deflector.
3. Wind deflector loses its integrated wire management capability.
4. Wind deflector is removed from the design altogether.

Figure 7.3.4 presents the composition structure of the PV racking assembly, with main inferences indicating the current state of the model in Protégé. On the top left, the asserted property relations of the PV racking (a1_pv_racking). As it can be seen, the wind deflector appears at the top of the list of proper parts. On the top right side, the figure shows both asserted and inferred types for the PV system. The asserted type is PV_racking_assembly, rendered at the top in bold letters. All the inferences are presented in Protégé graphic interface with a pale yellow background. The inferred types for the PV system includes all the aspect systems in which its parts play a functional role. Each aspect system corresponds to an environment-centric function that is at least satisfiable under the current state of the design model (i.e. Satisfiable_E_function).

The property assertions for the wind deflector are in the center left of the figure. It can be seen that the both slot and screw brackets are asserted as components of the wind deflector. Implicitly, they are also proper-parts of the wind deflector. The use of the has_component property was a necessary approach to avoid issues of applying cardinality restrictions on transitive properties, such as is the case for 'part-of' and 'proper-part-of' relations.

On the bottom right, the property assertions for the screw brackets (in the image only the left bracket is being shown). The red rectangle indicates that the bracket has two screws as fastener components. This allows the bracket to be classified by the reasoner as moment-resistant connection, which in turn leads to the inference of the wind deflector as a type of bracing device (red rectangle in the inferred types).

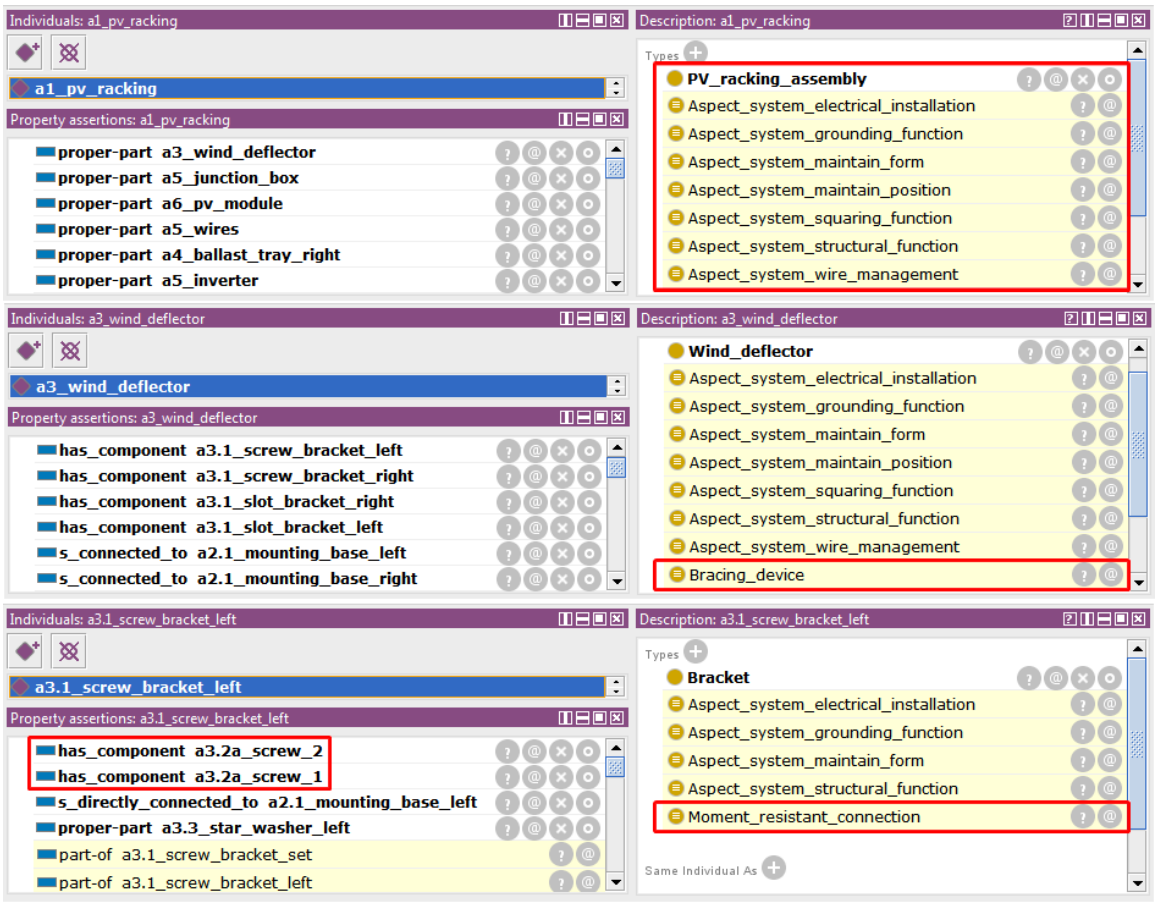

Figure 7.22: Initially complete state of PV system. Moment-resistant bracket allowing wind deflector to function as bracing device.

7.3.4.1 First design modification

The next Figure 7.3.4.1 presents the first design modification involving the removal of one screw from each screw bracket. This makes the bracket no longer a moment-resistant connection, which in turn leads to a loss of the bracing capabilities of the wind deflector. This is considered a pre-condition for resisting lateral loads and maintaining the overall structural form of the array.

Since that the nominal bracing capability disappeared, the high level function _fe7_maintain_structural_integrity is no longer realizable, and by consequence, the PV racking as a whole no longer appears as member of the corresponding aspect system. However, both

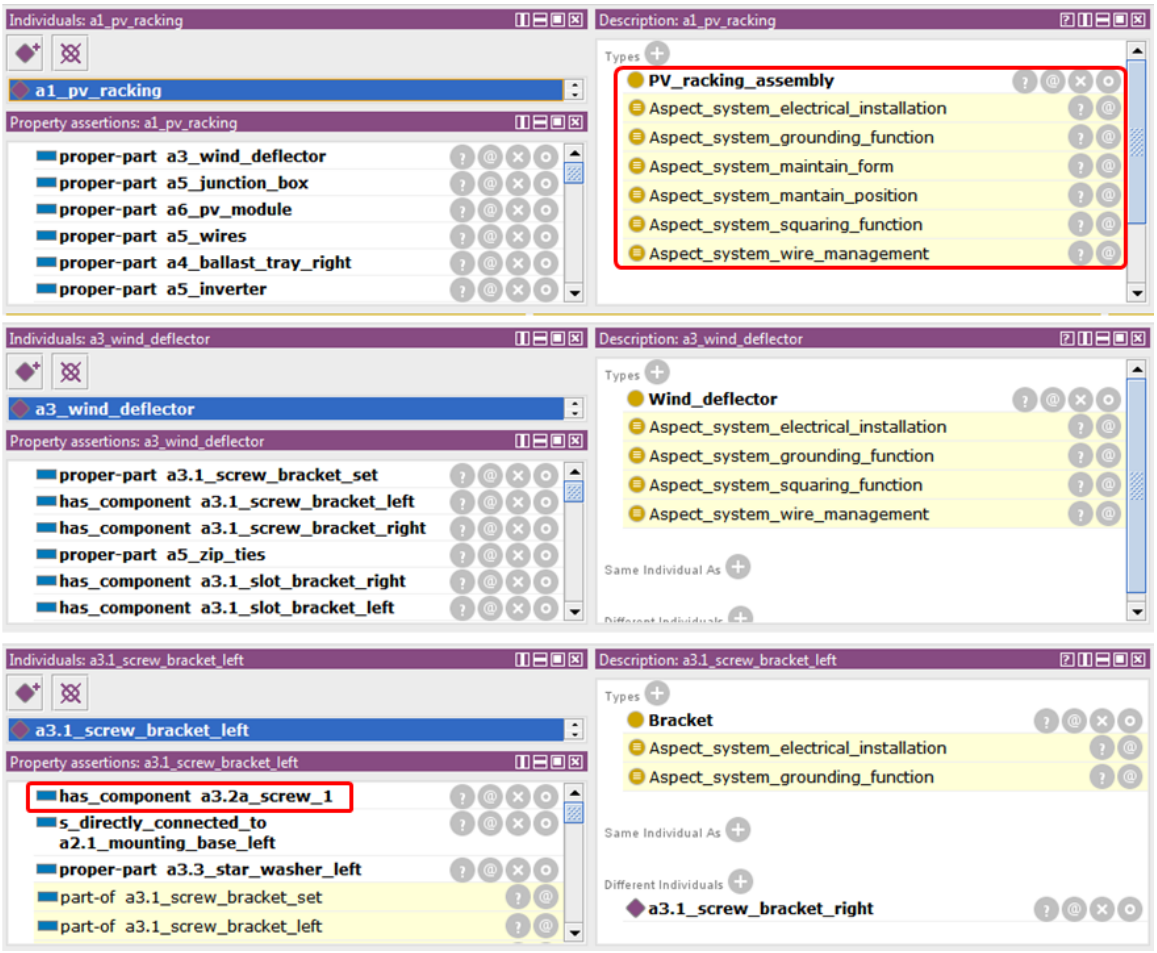

Figure 7.23: State of PV system model after first design modification. Screw brackets are no longer moment resistant, after the removal of one screw and wind deflector loses its bracing function. Overall structural integrity function is no longer realizable.

PV racking and wind deflector remain in the aspect systems for the sub-functions _fe7.1 _maintain_position and _fe7.2_maintain_form.

This means that the various parts of the PV racking, including the wind deflector and the screw brackets, still participate in the maintenance of form, but not by means of the qualified participation provided by moment resistance and bracing. This also means that maintenance of form is still somehow satisfiable without any device being nominally allocated to that function. In practice this may imply that the PV module itself is contributing to the bracing the array, which is not ideal but acceptable under certain conditions.

The fact that the wind deflector remains a participant in the maintenance of form and

position, and yet it is no longer a participant in the maintenance of structural integrity can be verified in the Figures 7.3.4.1 and 7.3.4.1. Both images show the combined inference of the aspect systems for these three functions, and explain in part the dependency among them. The inference is based on equivalent class expressions (i.e. DL Query), which returns the causal participants in the intended output perdurant of each function. The first figure presents the three aspect systems queried under the initial state of the design, without modification. The second figure in the following page presents the results of the same queries after the modification. Recall, that the modification is the simple removal of one screw from each screw bracket.

As it can be seen in the first query for the _fe7.2_maintain_form function before modification (center of Figure 7.3.4.1), the wind deflector, screw brackets and PV module are all co-participants, among many others parts, in the maintenance of form. However, only the wind deflector has a proper qualified participation, due to the fact that it is attached to the mounting supports with moment-resistant connections. After removing the screws from each screw bracket, these become simple pin connections. The wind deflector still participates in the maintenance of form, but this participation is not a nominal, qualified participation anymore. Thus, while this component is still classified as part of the aspect system for maintenance of form (center of Figure 7.3.4.1), it is not classified as part of the aspect system for overall structural integrity. In fact, the latter is returned totally empty, given that this function is no longer realizable nor satisfiable under the specified semantic constraints.

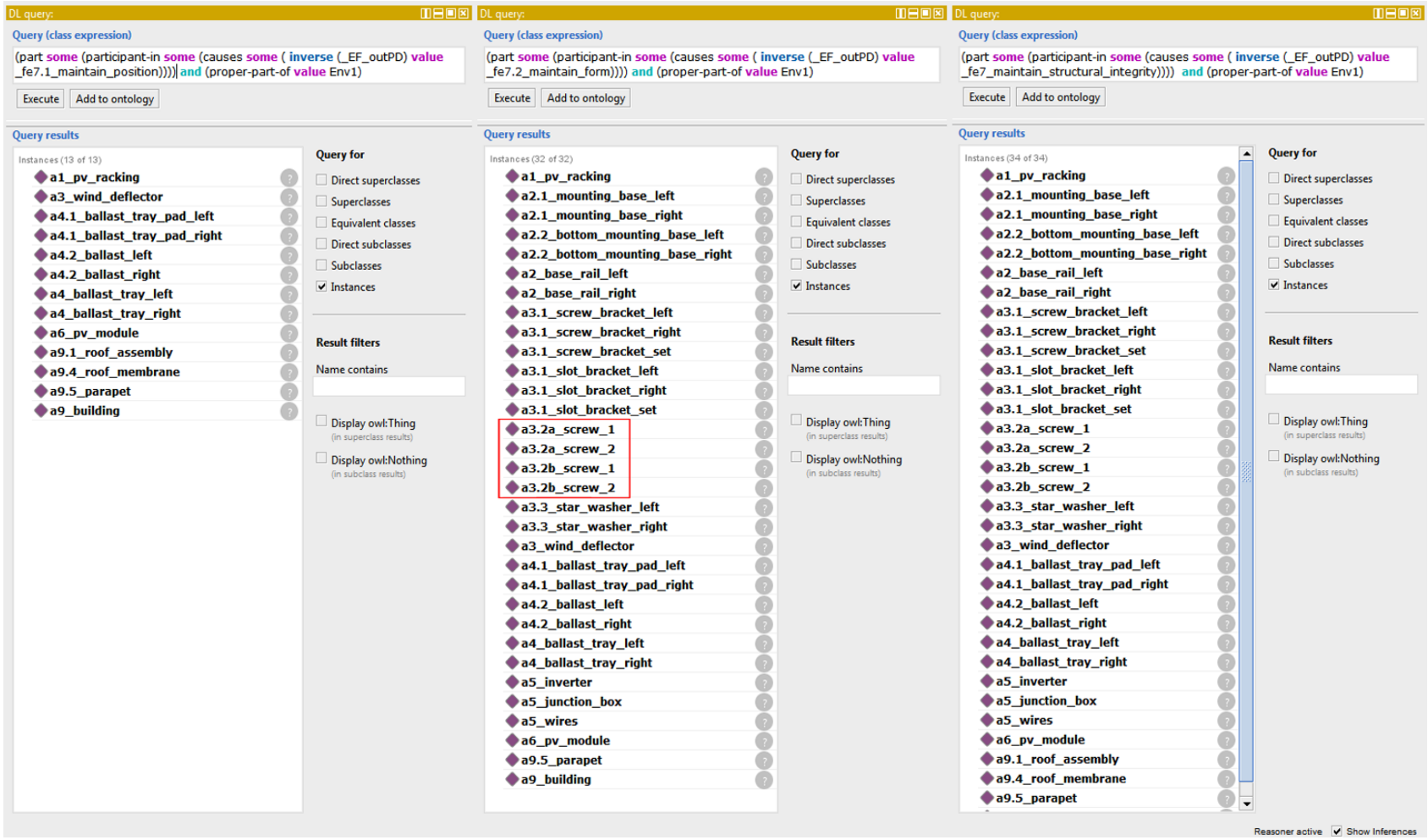

Figure 7.24: Aspect system for structural integrity. Cross-cutting dependency on moment-resistant screw brackets (before modification).

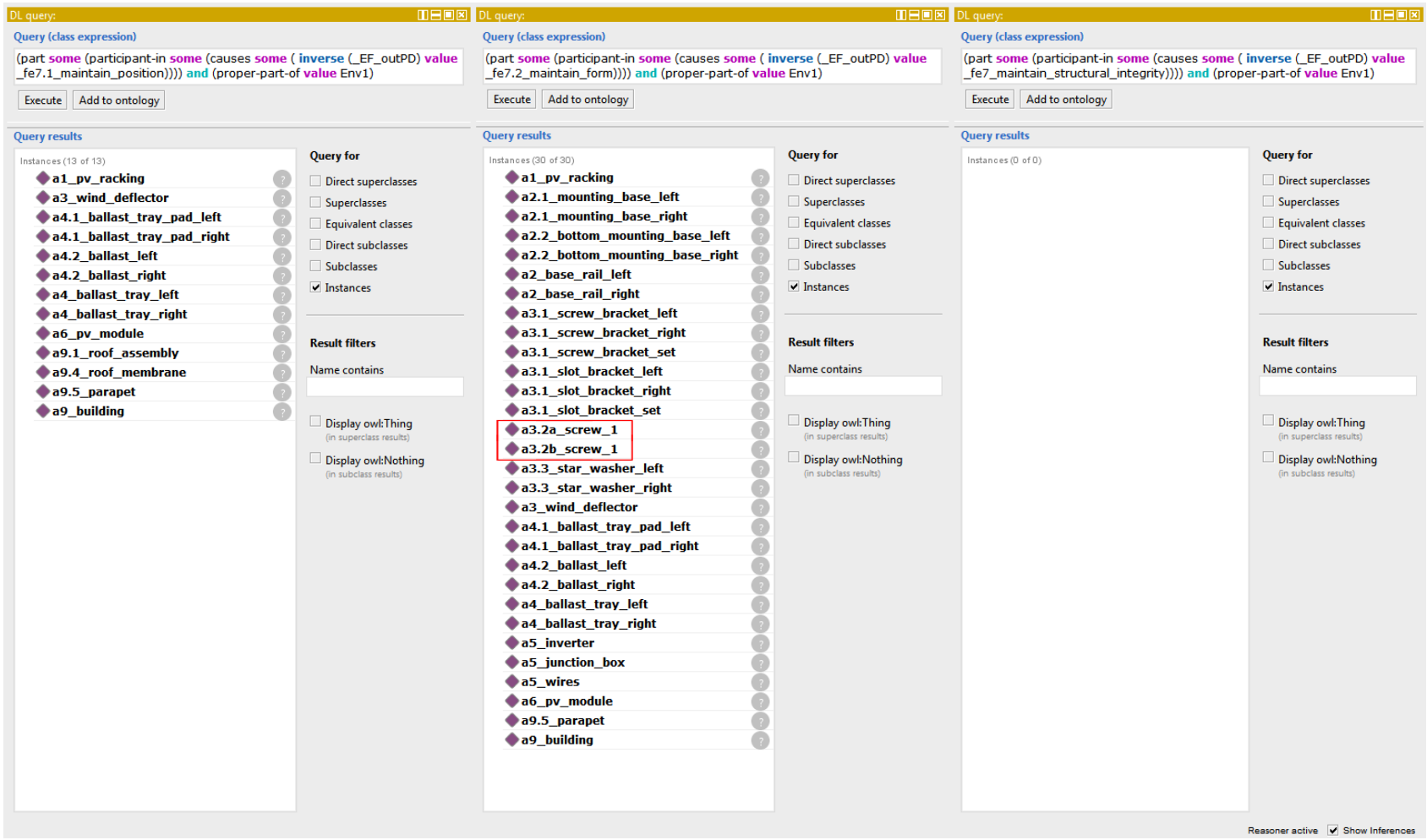

Figure 7.25: Cross-cutting implications of using only one screw per screw bracket. Structural integrity function no longer satisfiable.

To clarify the reasons for these inferences, it is necessary to present again the constraints imposed by the specification of the structural integrity function, introduced previously in listings 7.40 and 7.41. The formulation of these functions rely on a nested structure of dependencies, based on cardinality restrictions over equivalent classes defined at a lower level. Thus, structural integrity requires two entities of the class Maintaining_form_and_position as direct causal links. This class is the union of perdurants involved in the maintenance of position and form.

The occurrence of the latter in turn is dependent on the nominal, qualified participation of a bracing device, invoked internally by the input behavior b0 of the function _fe7.2 _maintain_form. If that link fails, as it does when no artifact is nominally allocated that function, then the cardinality restriction imposed by the structural integrity function can no longer be met, even though functions for the maintenance of form and position remain satisfiable. Another war to explain the inference processes, and the semantics leading to the current classification, is to request a query about the behavioral capabilities of each participant of the structural integrity function. This can be done with the following SPARQL query, done before and after the design change.

Listing 7.42: SPARQL query on behavioral capabilities of functional co-participants.

```
SELECT DISTINCT (?f AS ?e_function)(?o AS ?causal_link)(?b AS ?by_behavior)
           (?c AS ?constraint) (?v AS ?value)(?u AS ?unit)
WHERE
{  ?f a fr:Realizable_E_Function ;
      fr:_EF_outPD ?o .
   ?a a sol:Aspect_system_maintain_form ;
      fr:_B_has_capability ?b.
   ?b fr:behavior_constraint_on_perdurant ?o .
   OPTIONAL { ?c a owl:DatatypeProperty ;
              rdfs:subPropertyOf sol:has_behavioral_property ;
              rdfs:label ?u .
           ?b ?c ?v }
FILTER(?f!=sol:e_global) }
ORDER by ?f
```

The results returned for the design before the modification indicate that the wind deflector nominally participates in bracing. After the design change, the results still show the wind deflector as participant, but without the qualification, similarly to the way that a PV module also participates in bracing, but in an unintended manner.

Therefore, the distinction made by the reasoner is based on different types of participation, that is, a normal unqualified participation versus a nominal one. In a different scenario, the participation of the PV module or any other part in bracing could be qualified participation, including the addition of more specific behavioral constraints, such as moment of inertia, Young's module, or any other indicator necessary for analysis of structural performance. In such cases, the distinction could be made also based on specific parametric values. For instance, for values greater, equal or less than certain threshold, similarly to the model developed in the first case study.

7.3.4.2 Second design modification

The second design modification involves the complete removal of the screw brackets from the wind deflector, with a loss in electrical bonding for this component. As demonstrated in the previous design scenario, this has the advantage of simplifying of the installation process, because of the smaller number of parts and tools required. As result, the reasoner does not classify the wind deflector as a member of the aspect system of electrical grounding anymore (7.3.4.2). This change also implies that the wind deflector will have to bonded with the rest of the racking by other means.

In other words, localized bonding is a fairly device-specific function, unless all components of the system fail to provide electrical continuity. Therefore, while device-centric function _fd5_electrical_bonding is no longer classified as a Satisfiable_D_function , at a higher level of abstraction the environment function _fe5 .2 _electrical_grounding remains classified as a Realizable_E_function. According to the semantics of this framework, there would be no aspect system inferred for that function otherwise.

The table in Figure 7.3.4.2 summarizes the implications of these two design changes, by using the same query expression after each change. The query selects all members of the

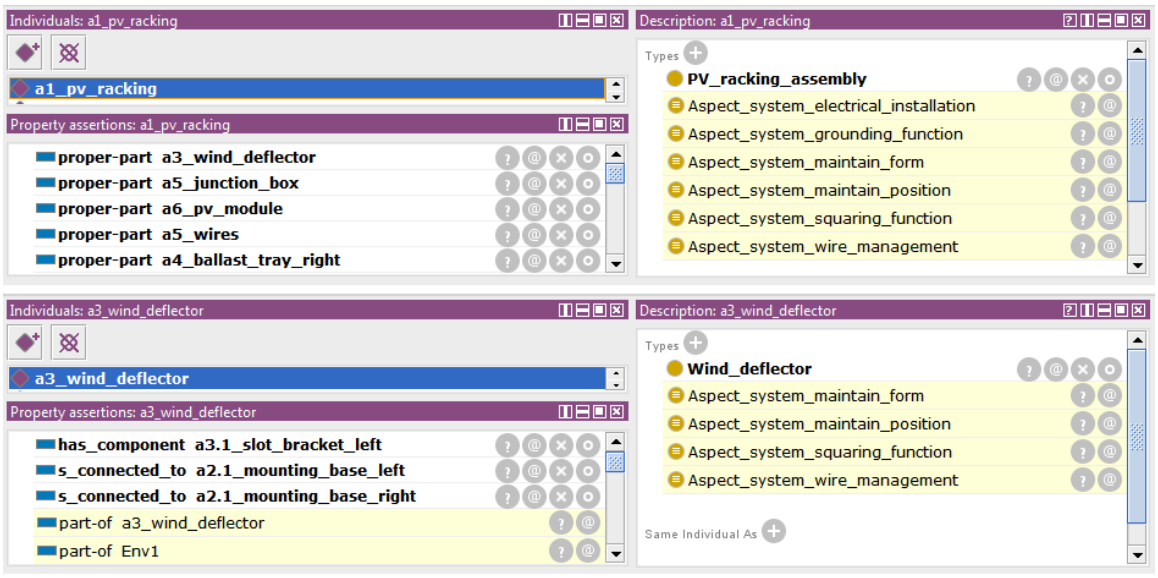

Figure 7.26: State of PV system model after second design modification. Both screw brackets are removed from wind deflector, losing its participation in the electrical grounding function, i.e. the wind deflector is no longer bonded to the rest ot the racking.

aspect system for the function _fe7.2_maintain_form, denoted by the variable ?aspect. At the top of the figure is the query expression, and the returned results for the complete state of the design, before any change. The returned list includes forty five entries, many of which are repeated because of the optional clauses added.

These optional clauses include all the behavioral capabilities of the members of the aspect system, denoted by the variable ?by_behavior. Thus, the member sol:a3.1 _screw_bracket_left has a moment resistant capability, and it is also electrically bonded by torque. The moment resistant capability is due to the fact that the brackets have more than one screw each, and therefore both are correctly classified as a moment-resistant connections by the reasoner. The electrical bonding capability is an example of aggregation of behavioral capabilities and constraints from internal parts done recursively. Since the star washer is a part of the bracket, all of its behavioral properties and values are added to the list of capabilities of the bracket as well. In this case, the bonding capability with an 'electrical resistivity' constraint of 20 milliohms.

Since the aggregation of behavioral capabilities is recursive, the top level of the racking

assembly aggregates them all. For this reason the individual sol:a1_pv_racking appears five times, once per each behavioral capability derived from its internal parts. This modeling functionality was considered useful to collect data about behavioral constraints and property within the same query of the aspect system, which could then be used as input by analysis applications distributed, possibly over the web.

Another advantage of this aggregation mechanism is that it allows to check quickly is the required capabilities are available in the system, without having to check for all the parts. A trivial but useful example is the tool box used for installation. By looking its aggregated capabilities, is easy to see if it has a tool with selective torque control (most likely an automatic screwdriver), or how much amperage the extension cord can take, assuming that they are still inside, as 'parts' of the tool box.

After the first change, the screw brackets have lost their moment resistant capabilities, due to the removal of the screws. This makes the wind deflector no longer act as bracing device, which in turn gets reflected in the individual a1_pv_racking itself, as it does not possess bonding nor bracing capabilities anymore. This can be seen in the central section of Figure 7.3.4.2, which shows the results after running the same query again. Now, only thirty three parts are included in the aspect system of function _fe7.2_maintain_form. Notice however that bonding by torque and electrical resistivity capabilities are still present, given that the screw brackets still keep one screw and star washer each.

Finally, after the second design modification, which involved the complete removal of the screw brackets, no bonding capabilities remain. Only the behavioral capability b_reduces_uplift_by_deflection of the wind deflector remains, with a coefficient of aerodynamic drag of 0.09.

The elimination of the wire management capability of the wind deflector, after the third design change cannot get reflected in this query because the expression selects only bonding, moment resistance, bracing and uplift resistance capabilities. A different query could be made very easily so to return all the internal capabilities of the system.

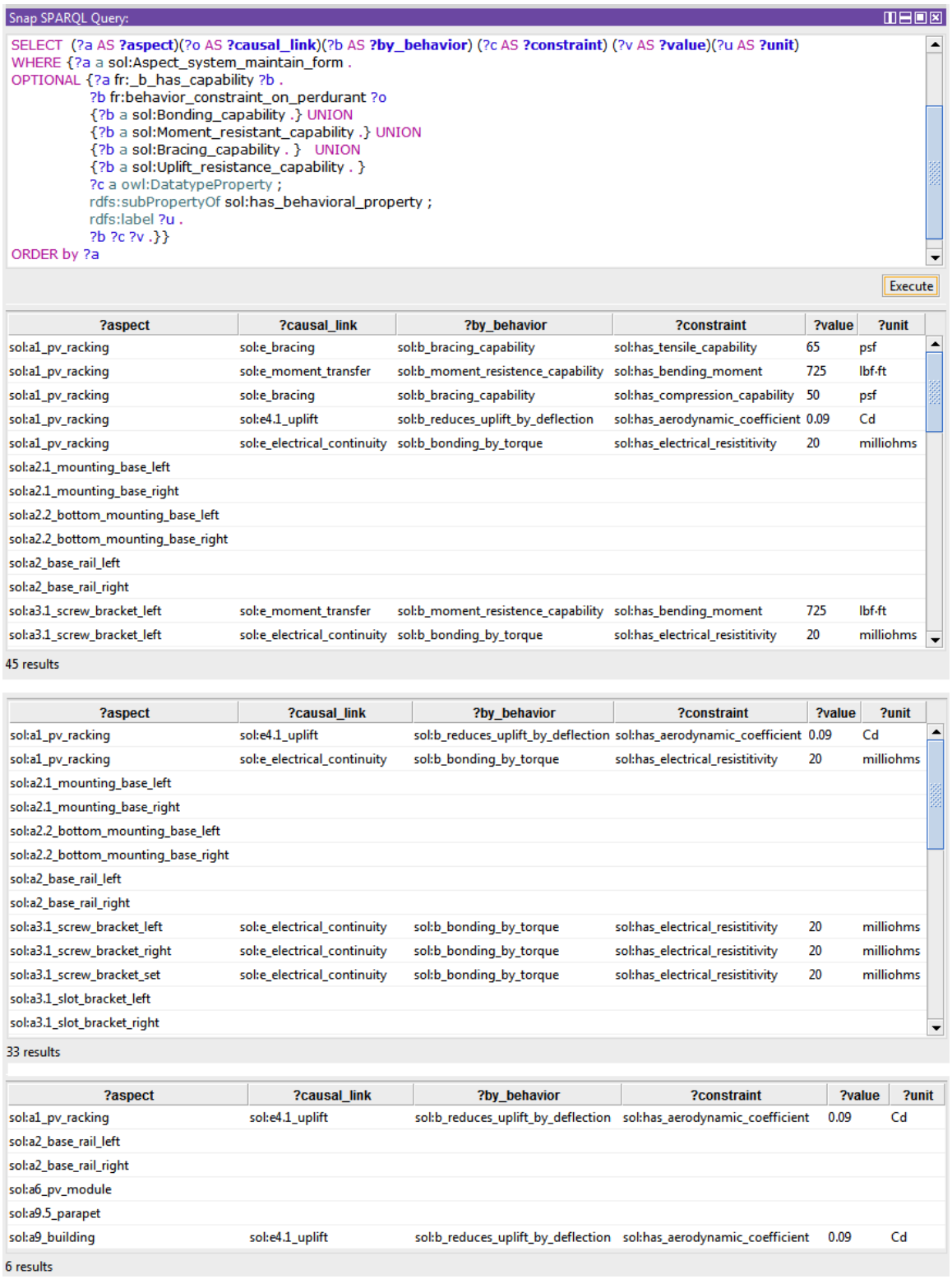

Figure 7.27: Evolution of high-level functional implications of low level design changes. Behavioral capabilities aggregated recursively at the top disappear after modifications.

7.3.4.3 Third design modification

The third design modification involves the removal of the integrated wire management function of the wind deflector. The practical implication is that electrical wires connecting rows of PV modules in series will need to be bundled and routed towards junction boxes and inverters by other means, while avoiding direct contact with the surface of the roof. These capabilities were originally allocated to the wind deflector in terms of a device-centric function. The allocation was made based on the assertion below.

Listing 7.43: Wire management device-centric function.

```
1 Individual: sol:_fd3_wire_management_D_function
2     Facts:
3       fr:_DF_artifact   sol:a3_wind_deflector,
4       fr:b0   sol:b_wiring,
5       fr:b1   sol:b_integrated_wire_support
```

By simply removing the reference to the wind deflector as nominal artifact, the device function is becomes unsatisfiable. This means that the function is downgraded to a generic behavioral pattern _P_causal_pattern, which is characterized by having a complete causal chain between input and output behaviors, while lacking a main nominal participant. From a en environment-centric functional perspective, wire management is still satisfiable, but not realizable in an integrated manner. In practice this lead to the need for using conventional zip ties, which are time consuming and prone to failure.

7.4 Discussion and results

The case study presented is fundamentally 'device-centric', containing about thirty class instances (10 endurants and 20 perdurants), and six asserted object properties. The main goal of this small model was the implementation of a minimum set of relationships necessary to describe the functional interactions discussed for the electronic device. These relationships, captured by the six object properties in OWL are: a) composition / parthood; b) participation; and c) causality.

Regarding the conditions in which the interactions of interest were to be inferred, these involved the modification of basic parametric values associated with structural properties of the design. In particular, modifications made were geometric in nature, related to dimensions of features of the casing. No connectivity or adjacency relations between components were defined, nor structural constraints regarding topology. Besides dimensional values, the description of the design structure was initially restricted to a single composition hierarchy.

The limitations of such incomplete structural description were identified during the development of scenario 3, where it became evident that the ventilation opening needed to be part of the ventilation subsystem as well. After asserting that relation, a different set of causal participation relations was obtained, making evident the need to capture *distal functional interactions* across components.

To address this problem it is necessary to realize that the opening is not a structural part of the ventilation subsystem, but a functional one. This is to say that the perdurants in which the opening participate are part of the main perdurants to be handled by the ventilation subsystem. In other words, the opening participates in a perdurant that is part of the *life* of the ventilation subsystem. However such participation is not essential to the existence of the opening. It can still exist as part of the casing without ever engaging in a process related with ventilation, such as it would be the case when the device is no longer working. This imposes the challenge of how to describe the structural connection between the opening and the ventilation subsystem under normal working conditions, without asserting a 'hard' parthood a-priori[4].

The characterization of functionality in terms of output perdurants, and participation relations without explicit behavioral qualification, i.e. behavioral constraints, may be advantageous in the development of simple models with small number of elements. However this approach is problematic given a more complex sets of functional requirements, such as those involved with functions from an environment-centric perspective. To address these issues, the use of additional structural constraints related with topological relations and

[4]In modeling languages such as SysML, a hard part-of relation is denoted by a solid black diamond in the top end of connector line, whereas a 'soft' part-of relation is denoted by an empty diamond.

multiple composition hierarchies was addressed in case studies 2 and 3. Moreover, the characterization of behavioral constraints and functional roles at multiple levels of abstraction was also addressed, including an extra case study presented in section 8.2 of Chapter VIII Conclusions.

For that reason, the complexity of corresponding models increased significantly, by qualifying participation relations with behavioral descriptors, also referred to in this research as behavioral capabilities. Device-centric functions and environment-centric functions were also introduced in latter case studies. Specifically, case study 2 addresses the relationship between these two levels in the context of designing a ballasted PV racking system for commercial flat roofs.

The third case study is based on an extension of the second one, addressing additional functional aspects of the PV racking system, including interactions with the functionality of the building roof. It also addresses the functional implications of design modifications in different phases of the product life-cycle, which illustrate scenarios of functional interactions that are distal in time. These modifications involve for the most part the addition and removal of features and components, providing a series of scenarios that demonstrate the capabilities of the proposed framework to support incremental elucidation of aspect systems.

In particular, the design scenarios developed in this case study allow to illustrate how the functional modeling approach adopted, and the inference capabilities embedded in the model, allow to identify new sets of unanticipated 'lurking' requirements, that emerge as result of last minute design decisions. The ability to describe and infer distal functional interactions in space were also explored in this case study, related with the multiple functional roles of the wind deflector design.

For each case study, incremental elucidation of aspect systems was demonstrated through a series of design scenarios, supported by different query methods, including DL Query and SPARQL. These scenarios were developed to assess the functional implications of design changes across different levels of abstraction and through various phases of the product life-cycle. In particular, the problem described in Chapter I, regarding the inference of functional implications caused by the removal of a multi-functional component (i.e. wind

deflector) was thoroughly demonstrated. Moreover, these queries allowed to validate the criteria of multi-functionality and functional integration of the research hypothesis, based on the inference of multiple functional participation relations of individual components.

The implementation of different types of composition relations, causal relations and participation relations in Aspecto-FR, based on the translation of a subset of DOLCE-FR axioms, allowed to demonstrate an approach to traverse a design model, supporting the identification of elements playing a functional role in the satisfaction of a function, and the selective 'harvesting' of behavioral constraints and capabilities that are relevant from a performance perspective.

Regarding the computational complexity of the models developed, and the performance obtained during reasoning and query tasks, a summary is provided in the following (table 32).

Table 32: DL metrics of 4 case studies. Reasoner (Pellet) and query runtime (SPARQL).

Metric	**Aspecto-FR**	**Case study 1**	**Case studies 2-3**	**Case study 4**
DL expressivity	$\mathcal{SRIF}$	$\mathcal{SROIF}(\mathcal{D})$	$\mathcal{SROIQ}(\mathcal{D})$	$\mathcal{SROIF}(\mathcal{D})$
Class count	73	99	177	106
Object property count	123	131	128	128
Data property count	-	9	73	5
Individual count	-	30	202	48
Class axioms				
Sub class axioms	106	128	238	147
Equivalent class axioms	4	14	26	17
Disjoint classes	18	18	19	18
Object property axioms				
Sub-properties	-	83	84	82
Inverse properties	46	46	46	46
Functional properties	1	1	1	1
Transitive properties	7	7	7	7
Symmetric properties	-	1	3	1
Reflexive properties	2	2	2	2
Property chains	16	17	17	17
Individual axioms				
Class assertions	-	3	89	15
Object property assertions	-	35	326	60
Data property assertions	-	2	56	3
SWRL rules	5	9	5	5
Performance				
Reasoner time* (ms)	168	3026	197489	5723
Query time** (ms)	1	3	141	13
Query time*** (ms)	-	10	326802	16537

CHAPTER VIII

CONCLUSION

The value of a building, or any other type of design artifact for that matter, ultimately lies on how well it enables the achievement of certain states of affairs that are needed or desired by a community of people. Within the chain of human needs, any given state of affairs are probably just a set of required preconditions for the achievement of even higher goals. Thus, in the context of the built environment, the provision of shelter, along with minimum levels of comfort and services are the most elementary preconditions to be met by any design project. By having basic needs satisfied, people can then engage with much more complex activities, eventually demanding additional functionality at increasing levels of sophistication.

Traditionally, the relationship between these multiple types and levels of functionality has been described in rather reductive terms, by reliance on the use of hierarchical tree-like representations, which tend to maintain a logical correspondence with the compositional structure of building technical systems. Unfortunately, this representational approach, strongly influenced by the so-called functional decomposition method, does not account for the semi-lattice structure of causal interactions and functional inter-dependencies underlying most of the socio-technical systems that are built, used and occupied by humans [8]. For this reason, attempts to provide computational models to represent, explain, predict, and ultimately design better integrated socio-technical systems, particularly those related to the built environment, have not been very successful so far.

To tackle this problem, it is important to realize that functional integration, in buildings at least, but possibly in other systems as well, has two different dimensions. The first has to deal with horizontal integration of functions allocated to components or subsystems at similar level of abstraction. For instance, the problem of integration involving envelope systems, such as curtain walls, with load-bearing structural systems. The second dimension

has to deal with vertical integration of multi-nested functionality, which are often very different from each other.

This vertical dimension has received considerably less attention than the horizontal one, until more recently with a growing body of work in research areas such as evidence-based design. Among the possible reasons is the fact that, while challenging from a technical viewpoint, horizontal integration is often made manageable by isolation from external considerations beyond those established by practice and convention. Standardization of products and methods contributes to create the illusion of isolation, given the labels of 'best practice' and certification from third parties, and sometimes even tradition within certain trades.

When it comes to vertical integration of functions however, conflict and uncertainty are less prone to fit normative or managerial approaches. In this case, one of the main design challenges is the integration systems of very different kind, especially because of the contextual nature of their functional interactions and inter-dependencies, which usually propagate of across multiple levels of the abstraction hierarchy. In the case of the built environment to be more specific, this can be translated into the problem of integration between so-called technical functions, usually a at a lower level of functional abstraction (e.g. device-centric functions), and high level 'soft' functions (e.g. environment-centric functions).

In the context of buildings, the relationship between indoor environmental conditions with physical and psychological well-being is a relevant example of this problem, which is receiving increasing attention from researchers of different fields. In particular, the impact of the built environment in health care, education and productivity have been subject to a growing number of studies, suggesting the need for a more comprehensive understanding of the different types of functionality and the causal relationships that lead to a better levels of performance, comfort and ultimately, to a better quality of life [307, 333].

This problem is certainly not unique to Building Design. Other design disciplines dealing with human factors, human-machine interaction and complex socio-technical systems also face a similar challenge. These may range from product development, both hardware and

software, to systems engineering, project management and organizational planning, to name a few. An increasingly common realization in many of these areas is that value proposition of any design or planning effort stems precisely from a successful vertical integration between 'hard' technical functions and 'soft' functional goals, usually at the top of the hierarchy. According to this point of view, even design innovation has to be driven by preferred outcomes at the highest level, rather than by meeting low-level technical requirements [196, 70].

Clearly this requires the representation of preferred outcomes in compatible functional terms, albeit at increasing levels of abstraction and (commonsensical) generality. Compatibility in this case involves, among other things, the mapping between the quantitative character of technical functions and the qualitative nature of many preferred outcomes [217].

This broader perspective on the issue of systems integration, including the integration of socio-technical systems, as they are increasingly ingrained within the built environment, defines primary motivation of this research. At a more specific level, the interest is in the development of computational models capable of providing effective support for the design of better integrated building systems.

8.1 The theoretical relevance of aspect systems

The research specifically addresses the problem of providing a formal characterization of building functions and associated concepts. Altogether, the integrated formalization of structure, function, behavior should provide the basis for a operationally robust functional modeling framework that addresses the specific requirements of the domain. Among these, the research has identified the need to support the description of multiple functional viewpoints, at different levels of abstraction, regarding granularity, as well as different phases of the building life-cycle.

For that purpose the concept of aspect systems is considered a fundamental abstraction, to satisfy these requirements. Therefore, an formal characterization of aspect system is needed, which constitute the main objective of this research. The functional viewpoints, and related set of requirements to be supported by a formal characterization of aspect

system are generalized in this research as follows:

- **Distal functional dependencies:** Two additional functional perspectives required have to deal with how entities that are distant from each other in space or time are described as functionally related. In the first case, a structural relation needs to exist, but for which there is no explicit description in the model. This problem relates with the topology of physical connections, which in general are only described if the nature of functional interactions are known beforehand. The second case relates with the functional implications of past events in the state of affairs of the present. However, during design, the relevance of this type of relations stems from situations where design decisions made to improve performance aspects of a specific life-cycle requirement, have unanticipated consequences in early or late life-cycle requirements.

- **Supply and demand:** A common representation of functions that serve both, the specification of required functions, before or during design and construction (demand side), and the specification of functions provided, after design, fabrication or construction (supply side).

- **Multiple levels of abstraction:** Inference of aspect systems require the identification of functional interactions across different sub-systems and levels of abstraction. This implies the need for a general, compatible representation for the different, domain-specific meanings of function used in building design. Such compatibility has to support integration in two orthogonal directions.

  - **Horizontal integration (domain-specific functional meaning):** Integration of domain-specific meaning of functions associated to different technical sub-systems at the same level of aggregation (e.g. electrical and mechanical subsystems, etc.).

  - **Vertical integration (hard and soft functions):** Integration of functional meaning across different levels of abstraction. This involves the problem of how the semantic content of low-level functions, usually technical and quantifiable

(e.g. heat pumps), relate with the semantics of intermediate and high-level functions, which usually are not technical or easily quantifiable (e.g. productivity).

- **Nominal and anonymous functional roles:** This class of functional viewpoints relate to the fact that many parts of buildings perform more than one main nominal function. At the same time, many functions are affected by the participation of multiple parts, either negatively or positively. The set of these parts is the aspect system of the function, and the specific goal of this research is precisely to enable the identification of these parts. However, not all elements in a BIM model, if any, contain explicit descriptions of heir main nominal function, let alone additional, and often idiosyncratic functions. Any additional functional role performed, either intended by design or accidental, is either implicit in the model or unknown. Therefore, the identification of the members of an aspect system requires the explicit characterization of these additional functional roles. For that purpose, the formalization of various functional and behavioral aspects of building elements is required, so that all relevant functional roles can be identified by means of inference.

The following two subsections provide a discussion of these requirements. First the most theoretical, and arguably difficult issue of representing distal relations is addressed. This is done with some examples provided from case study 2. The rest of the requirements will be discussed in the context of case study 4, which provides a minimal working example, introduced here to guide the discussion according to more specific aspects of the implementation.

8.1.1 Distal relations

In principle, the notions of distality and proximity could be applied for spatial as well as temporal relations. Both are relevant for the process elucidation of aspect systems, because the interactions and side-effects may span different phases of a product's life cycle.

One example of distal temporal relations is given in the second case study (design scenario 2), where the decision of using screw brackets to satisfy the electrical grounding function associated with the operations and maintenance (O&M) phase of a PV system,

changes the aspect systems of functions associated with the installation phase. Namely, the hardware installation function and the electrical installation function, both constrained in terms of number of assembly steps, time per step, part and tool count, and other ergonomic considerations with impact on productivity. To improve this process, the original design was conceived to be assembled without the need of any tool. However, the addition of screw brackets triggers a series of additional requirements, such as the need for power tools, which changes the composition of the aspect system associated with installation performance.

In this model, participation relations are not indexed by time (i.e $PC(a, e, t_1)$), so that temporality is not explicit (e.g. $t_1 < t_2$). Instead, the concept of behavior environment is used as abstraction for time. Given that the installation environment is different from the O&M environment, then it is possible to say that functional interactions are distal in time. Clearly, this is not entirely satisfactory, unless such temporal characterization is made explicit in the specification of the corresponding environments, which is not the case in the model yet. Future work will need to address this limitation. However, the use of behavior environment as temporal abstraction provides a simple and effective method to elucidate the implications of design changes across different periods of time. The figure below shows the eight different aspect systems defined for the PV system, classified according to the installation and O&M environments, labelled Env1 and Env2 respectively.

Regarding distality of spatial relations, perhaps it is necessary to make a distinction between distality regarding spatial extension, measured by length, and distality regarding degrees of separation, which is essentially a topological relation of connected parts. For instance, two objects might be spatially very close to each other, but separated by multiple layers of other stuff, so that *degrees* of participation and causality that would normally apply, do not. Masonry walls are good examples of this. For instance, multiple-wythe masonry walls and single wythe wall perform very differently regarding process of heat and mass transfer. While the former imposes many more layers of insulation against heat and moisture migration, these processes still occur, albeit at much lower rate than the latter, to the point that they might be irrelevant.

The fact that transfer still occur is because the multiple layers of the wall are *connected*

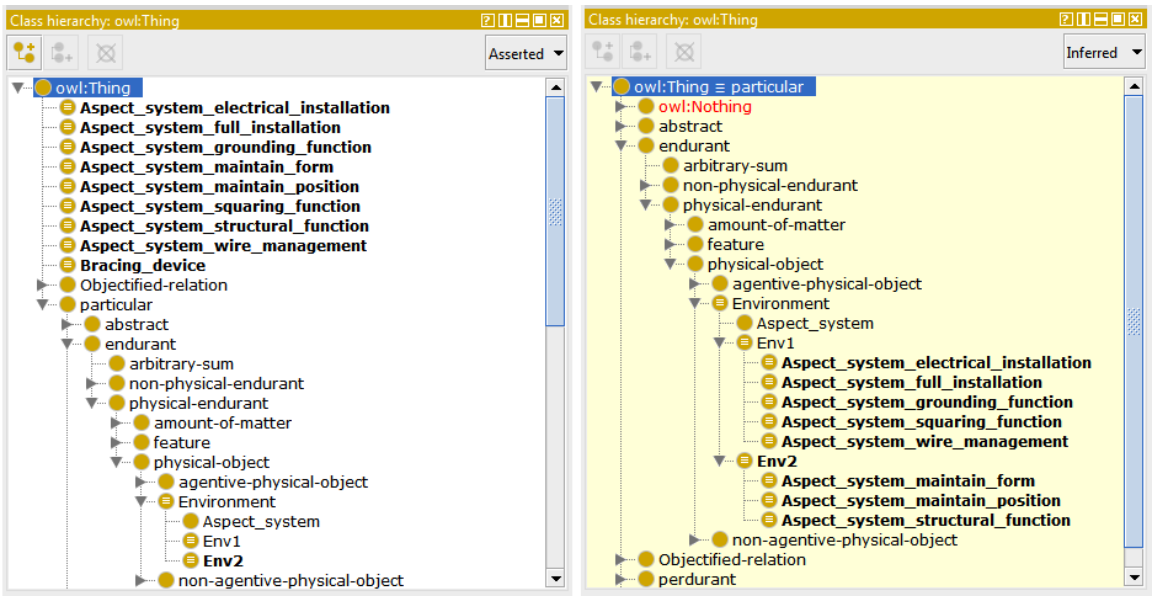

Figure 8.1: Behavior environments as abstraction for distal relations in time. On the left, eight different aspect systems asserted in terms of equivalent classes. On the right, inferred classification of aspect systems under different environments (Env1 and Env2).

somehow, even when cavities are present. The reduction of transfer rates is due to the properties of each layer of the wall, including material properties and thicknesses. The thicker the layer, higher the insulation effect, since the 'microscopic' distance has increased. For cavities these two properties also apply, since they are not really empty after all. This means that the spatial region occupied by air also needs to be included as a *connected* part of the wall assembly, with its own material properties and thickness.

This observation can be generalized in the model, with the use of the transitive relation s_connected_to, and its functional (in the OWL-DL sense) sub-property s_directly_connected_to . These relations, along with the explicit assertion that volumes of air are physical endurants subject to composition and participation relations, provide a basic vocabulary to represent distal relations both in terms spatial extension, as well as in terms of degrees of separation.

It is however the task of domain-specific knowledge bases to establish the rules and conditions for participation and causality associated with air, or any other medium relevant to the description of flows or physical transfer processes, such as heat, mass or momentum.

Further generalizations could include conditions for information transfer as well, using similar modeling patterns.

The reference of air as medium for heat transfer was preliminarily addressed in the first case study, but the specific type of participation was not made explicit. Instead, simple parametric relations were used to convey its participation implicitly. A more concrete, short example of explicit participation of air with spatial distality will be given in the next subsection addressing causal relations. This will allow to better explain the approach envisioned in this research to address this problem.

8.2 Minimal working example: failure mode in a co-working space

The section presents a small case study to demonstrate the main modeling features and capabilities of Aspecto-FR. The problem is based on previous examples related with multi-functional open office spaces, which are becoming increasingly popular in large cities under the business model of co-working spaces. In this type of spaces, a variety of amenities are offered to tenants to promote socialization and a stronger sense of community within the working environment. These may include from open kitchens and bars, to ping pong tables, lounge areas, gymnasiums, etc. Indoor plants are also common for various reasons, from infusing a sense of informality, to physical and mental health benefits.

The problem however is that the larger the number of services and activities, the more complex the web of behavioral interactions and side-effects that could lead to potential functional conflicts. The purpose of the functional model built for this example is not to capture all these interactions, but to demonstrate how one particular kind of problematic interactions could be described under the proposed framework.

In particular, the model of the co-working space exemplifies how the definitions of device and environment functions could be used to describe possible *failure modes*. Given that a failure mode is essentially a description of the conditions leading to an effect on the environment, the same modeling principles could be applied, including the description of behavior environments, modes of deployment and aspect systems associated to a failure. The only conceptual difference with a conventional function is that the resulting effect

is usually undesirable under normal circumstances. In this case, the undesirable effect is condensation of internal surfaces of the space, including the glazing of windows and curtain walls, due to excess of moisture in the indoor, as and temperature difference between indoor and outdoor.

With this example, the implementation of the requirement items for the framework will be demonstrated, including intensional definitions for teleological patterns, and nesting of functions. The example will also be used to provide a more detailed account of the approach adopted for the representation of distal relations, discussed earlier. Cross references will be provided when appropriate to previous chapters and sections for more detailed information.

8.2.1 Initial asserted model

The structural model for this case is fairly simple, relying on a basic composition of part instances, as indicated in table 33, which also includes the assertion of types for some of the individuals.

Table 33: Nominal participation for wind deflector with behavioral qualification.

?part	?type	?by_behavior	?participation
a1_office_building	Technical_artifact	-	-
a1.5_workers	Animal	-	-
a1.4_indoor_air	-	-	-
a1.3_curtain_wall	Technical_artifact	-	-
a1.2_open_kitchen	Technical_artifact	-	-
a1.2.1_boiler	Technical_artifact	-	-
a1.2.1_boiler	Boiler	-	-
a1.1_office_space	Technical_artifact	-	-
a1.1.1_indoor_plants	Vegetal	-	-
a0_outdoor_space	-	-	-
a0.1_outdoor_air	-	-	-

Implicitly however, some participation relations have been assigned for living entities, such as office workers and indoor plants. The listing below provides an overview which states that all living entities participate in some form of metabolic process, which in turn is defined simply as a perdurant that directly causes transpiration, denoted by _d_causes. The complete model is provided in the Appendix F.

Listing 8.1: Functional specification used to represent a failure mode.

```
Class: cow:Living
    SubClassOf:
        dc:agentive−physical−object,
            dc:participant−in some cow:Metabolism
Class: Metabolism
    SubClassOf:
            dc:process,
            fr:_d_causes value cow:e_transpiration
```

This type of cause, along with the general approach adopted in this research for the representation of causal relations, was discussed in the section (6.3.3), of Chapter VI. The discussion also include some of the issues and limitations of these approach regarding qualified participation. Admittedly, the topic of causality is a difficult one, and a rigorous ontological treatment is beyond the scope of this research. While future work will have to address this issue more in depth, the preliminary approach adopted in this research has been effective to fulfill the basic requirements for the implementation of a proof-of-concept framework. The following subsections will describe and discuss the relevant aspects of this implementation.

8.2.2 Participation and causality

The implementation of the notions of participation and causality, as derived from DOLCE-FR, was based on theoretical as well as pragmatic considerations. From the theoretical side, the intent was to provide a general characterization aligned with the commonsense use of these terms, especially of causality, in design fields, which admittedly, may differ from the specific meaning adopted in scientific fields. On the pragmatic side, the implementation approach adopted is guided by the modeling constraints related with the OWL-DL language, and the capability of the reasoners available, in this case, Pellet [283].

In this context, the object property _d_causes stands for a *direct cause*, and it is characterized as a functional property in OWL-DL. This means it can only take one single value, whereas its super-property _c_causes, which stands for a *contributive cause*, is characterize

as transitive, and therefore can have multiple values.

This choice corresponds to a modeling pattern in OWL-DL, used for instance in the implementation of linked lists [96], allowing to have more control over the selection of relationships of similar type, which is also critical during the reasoning process, given the problems related with the processing of complex transitive properties.

Regarding the functional meaning of these two causal relations within the theory proposed, a direct cause is taken as a necessary event for a change of state, whereas a contributive cause is not necessary, but as the name says, may contribute to a change of state. This idea underpins the process of inference over interactions and side-effects required for the elucidation of aspect systems, as proposed in this research.

Beyond aspects of implementation, the notion of contributive cause allows to establish a conceptual framework to reconcile, at a general level at least, the different meanings of function that need representation in Building Design. In particular, the intent is to support vertical integration of functional perspectives, between low-level, device-centric and high-level environment-centric perspectives, where the former tend to be more technical and domain-specific in nature than the latter.

Thus, the deduction of causal participation of an artifact in an output perdurant of a function, either by intention or by side-effect, allows the inference of the *role* played by such artifact in the satisfaction of the function. In this way, the interpretation of *function as a role*, which has been suggested as the most general, and therefore suitable for the domain of engineering design, in the discussion offered at the end of Chapter IV (section 4.3.4), and in the introduction of Chapter V (5) , could be operationally implemented, within the context of Building Design.

Figure 8.2.2 illustrates the relationship between participation (simple or nominal), direct and contributive causes, and causal participation, which supports not only propagation of functional roles at various level of abstraction, but also the description of distal relations.

In Aspecto-FR, the relation of causal participation is a derived object property, obtained automatically by means of a property chain, also presented in the Chapter VI, section 6.3.3. In the context of the example, this property enables the inference of the causal participation

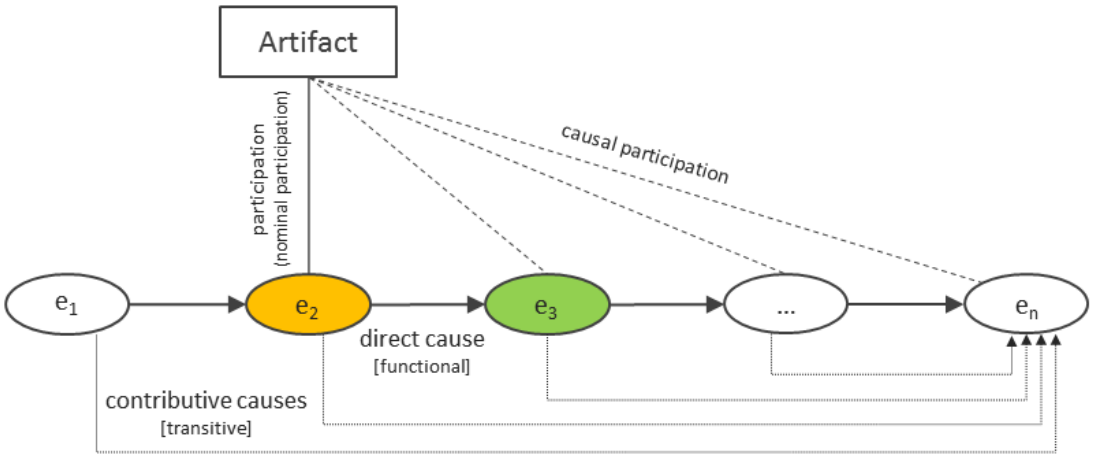

Figure 8.2: Direct and contributive causal relations between perdurants, and two basic types of participation relations, obtained by the functional and transitive characteristics of the causes, as defined in OWL.

of anything inside the co-working space that produces evaporation, in the causal chain leading to the process of moisture transfer, and eventually to condensation of surfaces, such as the glazing of curtain walls. This causal chain, called a causal model, is only activated however if the right combination of structural and behavioral constraints are in place.

8.2.3 Constraints and interaction patterns

Different constraints apply for either the category of endurants or perdurants, and in the model they are specified in chunks, or teleological patterns involving conditions of participation and causality. The most basic constraint for endurants, following DOLCE-FR, is that a participant needs to be part of the behavior environment. A second condition, introduced in the development of case study 2 7.2, and further developed in case study 3 7.3, is the condition of structural connectivity to a part of such environment. While in the case studies, that condition applies to describe mechanic connections such as fasteners, brackets and bracing devices, in this example it is used to denote physical contact with air. This allows distal relations to be represented in an abstract, but simple and effective manner by reusing the same type of relational object property. Further characterization of the type of 'connection' with air can be obtained by the characterization of specific types of

participation in which air is involved, particularly those associated with various processes of physical transfer.

In this case study, indoor air can only play a functional role in the in the 'satisfaction' of the 'intended failure', if it is in direct contact with entities that produce moisture, either by transpiration or by evaporation (e.g.a1.2.1_boiler.). In other words, air can only transfer moisture if it has absorbed moisture in first place. The entire physical process is certainly more complex, but its description is kept here as simple as possible for illustrative purposes.

The combined specification of structural and behavioral constraints in this case is made by the definition of the class Humid_air, a subclass of air. As it can be seen in Figure 8.2.3, this class is defined by an equivalent axiom that specifies that humid air is has to be part of the building, i.e. indoor air, and it has to be in contact with some living entity (e.g. plants or animals). If an instance of 'indoor air' fulfills this conditions, then it is considered to be humid air. It is important to point out that the degree of humidity, or more specifically, the amount of water vapor in the air, as measured in terms of relative humidity, is not relevant at this level. Further conditions can be applied later to refine the model.

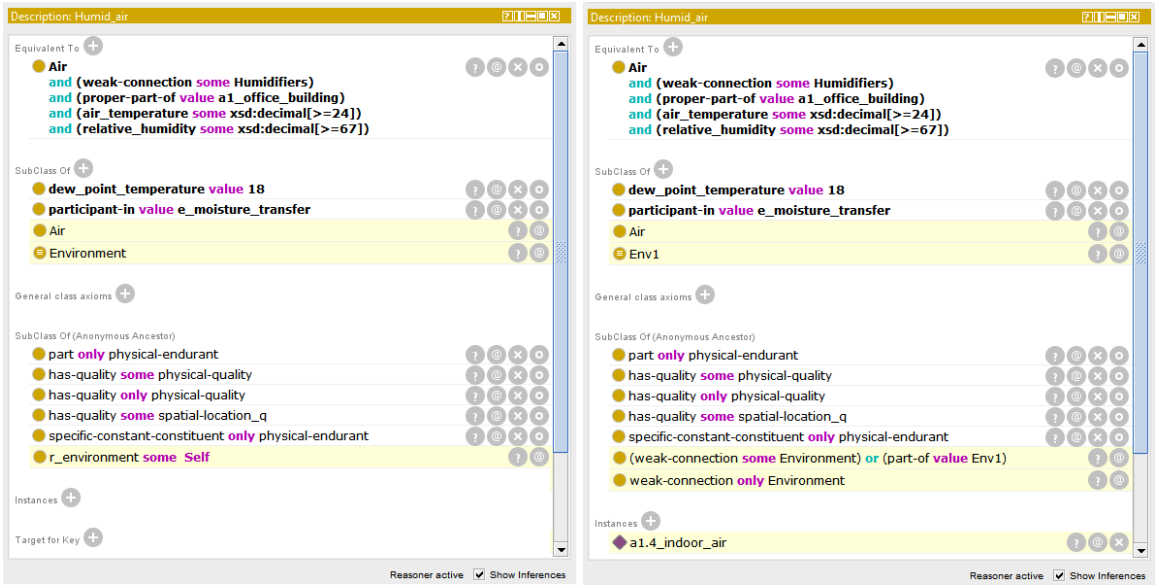

Figure 8.3: Equivalent class axiom for inference of humid air. Inferred individual is said to participate in the process of moisture transfer.

This axiom specifies that once a particular instance of 'air', that is, a volumetric body of air that is identified as part of a particular building zone or space (e.g. *IfcZone*, *IfcSpace*), becomes known as being in contact with an entity that releases moisture, then it follows that such body-of-air instance participates in moisture transfer. In the left side of the figure, no instance has been classified because no 'humidifier' entity was known to be in contact with indoor air (proper-part-of building). This could have various interpretations, but most likely due to the fact that the space is not occupied, or 'humidifiers' have been isolated from direct contact with indoor air. After contact was asserted, by means of the property s_directly_connected_to, the individual a1.4_indoor_air is then inferred as a 'humid air'.

To generalize all instances that produce and release moisture into air, the class of Humidifier has been defined, in terms of all entities with *causal participation* in moisture absorption. Since a causal participation is not a direct participation to be asserted, but an indirect participation to be inferred based on a property chain, it specifies an indirect effect in the environment caused by any of a form of evaporation, including transpiration or water that has been heated or boiled in some way. Hence, while a humidifier is known to release moisture into the air, i.e. its main participation, it is by the effect of some medium absorbing the moisture, that such participation becomes relevant from an environment-centric perspective.

A third class to endurants required to characterize the process of condensation as environment-centric 'failure' is the class of all surfaces that actually may suffer from condensation, called here a Condensable_surface. In the model, the process of condensation has been simplified for illustration purposes. Thus, to be 'condensable', a surface must be directly connected to other two entities. The first being a medium (i.e. indoor air) participating in moisture transfer, an having a dew point temperature greater than 16 Celsius. This parameter results from particular combinations of temperature and relative humidity specified in the model. The second entity is outdoor air, which has a temperature less than 24 Celsius. This contributes to a temperature of the surface that is less than the dew point temperature of indoor air, eventually leading to state of condensation.

The following listing provides the formal definitions of these three classes of participant

endurants, based on equivalent and subclass axioms that specify the set of structural and behavioral constraints used in this model. Further characterization could be provided using this same vocabulary to capture a more appropriate set of domain-specific rules and constraints.

Listing 8.2: Interaction patterns and causal model for condensation.

```
Class: cow:Humid_Air
    EquivalentTo:
        cow:Air
        and (dc:weak−connection some cow:Humidifier)
        and (dc:proper−part−of value cow:a1_office_building)
        and (cow:air_temperature some xsd:decimal[>=24])
        and (cow:relative_humidity some xsd:decimal[>=67])
    SubClassOf:
        dc:participant−in value cow:e_moisture_transfer,
        cow:dew_point_temperature value 18
Class: cow:Humidifiers
    EquivalentTo:
        fr:causal_participant_in value cow:e_moisture_absortion
Class: cow:Condensable_surface
    EquivalentTo:
        (fr:s_directly_connected_to some
            ((dc:participant−in value cow:e_moisture_transfer)
             and (cow:dew_point_temperature some xsd:decimal[> 16])))
         and (fr:s_directly_connected_to some (cow:air_temperature some xsd:decimal[< 20]))
    SubClassOf:
        dc:participant−in value cow:e_moisture_retention
Individual: cow:e_moisture_retention
   Facts:
      fr:_c_causes value cow:e_condensation
```

The method used for the specification of structural and behavioral constraints to define classes of endurants, can also be applied to define classes of perdurants. While in the former,

the specification leads to the inference of participation relations, in the latter constraints are used to support the inference of causal relations between perdurants, which conform a specific causal model for a mode of deployment satisfying the function or, conversely, the failure.

The formal definitions in OWL-DL for moisture transfer and condensation in the next listing exemplify the implementation of this proof-of-concept, and the specific interpretation made in this research of the notions of causality and mode of deployment from DOLCE-FR, introduced in Chapter VI (section 6.3).

Listing 8.3: Moisture transfer axiom. Constraints for teleological patterns.

```
1  Class: cow:Moisture_transfer
2      EquivalentTo:
3          dc:participant some (dc:weak−connection some cow:Humidifiers)
4      SubClassOf:
5          fr:CauseMD,
6          fr:_c_caused_by value cow:e_moisture_absortion
7  Class: cow:Condensation
8      EquivalentTo:
9          dc:participant some cow:Condensable_surface
10     SubClassOf:
11         fr:CauseMD,
12         fr:_c_caused_by value cow:e_moisture_transfer
```

These small number of axioms are sufficient to describe the basic set of conditions required for condensation to occur given the simple functional model provided. These conditions ae specified in the form structural and behavioral constraints over classes of both endurants and perdurants. Once proper instantiation takes place in the model, the correct inferences are made by the reasoner regarding causality and participants.

However, in order to make such definitions operational, in terms of modularization and reusability, they need to be encapsulated in some sort of intermediate abstraction that captures the context of the failure being described. In Aspecto-FR, this is done in

two complementary levels, namely, a device-centric and an environment-centric functional perspectives.

In the device-centric perspective, the triggering failure is the existence of 'humidifiers' in contact with indoor air. Because the distinction between failure and function is extrinsic to the phenomenon, and rather arbitrary, the same formalization made for a device function $DevFunc(a, b_0, b_1)$ proposed by DOLCE-FR (in 23), and translated into OWL-DL (in 6.3.7) can be used.

The only difficulty however is that, in order to be useful during reasoning, the specification of a device function at the instance level needs to be satisfiable first, as explained during the presentation of the second case study (in 7.2). This involves two conditions; first that there is a complete causal chain for the function (i.e. b_0 leads to b_1); and second, that a nominal artifact a for the function is known.

The problem is that this last condition is exactly what is missing in the formulation of a failure mode. After all, if the 'culprits' for a particular failure were known beforehand, all the modeling and reasoning efforts would unnecessary. The approach developed in this research to deal with this problem is presented in the next subsection.

8.2.4 Daemons, daemon functions and daemon patterns

The concept of a *genius loci* in ancient Rome was used to refer to the protective spirit of a place, and from which a place acquired its peculiar character, or particular pattern of being. This is somewhat related to the Greek concept of *daemon*, which refers to nature deities or spirits considered generally as benign. More recently, the term 'daemon' has been used in the context of computer operating systems as a program that provides internal services without any direct control of an end-user.

The underlying ideas for both concepts have received considerable attention in Architectural theory, especially after the work of Christian Norberg-Schulz in the phenomenology of daily environments [240]. Moreover, it can be argued that much of the influential work of Christopher Alexander in Pattern Language is grounded on a similar set of ideas [7].

For all these reasons, the term *daemon* is adopted here to refer to the class of anonymous

entities playing a main functional role in the achievement of a function, even if such function is considered 'undesirable' under normal circumstances, in which case it could be considered an *erroneous function*, a type of *failure mode* [260].

The concepts of 'genius loci' and 'daemon' are introduced here as analogy for interaction patterns that are fundamentally teleological, and allow to exemplify the approach adopted in this research for the definition of design elements by *intension*, in terms of such patterns. An initial discussion on this topic was given as part of the review of the IFC data model, in Chapter III (3), specifically in section 3.3.3. Among the main limitations identified in the semantic characterization of IFC, was its almost complete reliance on the use of extensional definitions.

A main problem of such method of semantic characterization is that it requires explicit 'typing' of members of a class. For example, in a BIM application, a moment-resistant connection would need to be explicitly declared as such, either during instantiation or afterwards. This is problematic for various reasons, but mostly because it assumes that any given part or component has to fit neatly within a fixed, predetermined taxonomy of functions. This imposes an artificial barrier in the processes of design, both in terms of creativity and collaboration, and it is one of the main limitations of the 'kit-of-parts' modeling approach underlying several of the early integrated design environments analyzed in Chapter II (2).

A fixed set of functions presupposes a fix set of behavioral aspects to be described, and by consequence, a fixed set of structural configurations available in the solution space of a design problem. It also presupposes a fix set of performance requirements, analysis applications and workflow scenarios that may not abide to real-world constraints of practice. These limitations are particularly problematic when it comes to situations that require complex integration of multi-functional elements that do not conform to clear-cut classification systems. A trivial example of this problem was given in the discussion about 'architectural' and 'load-bearing' wall, also in Chapter III (in section 3.4).

Part of the issue stems from the inability of rigid categorizations to support multiple functional views at different levels of abstraction. Some efforts in the AEC industry have

tried to overcome this problem by developing comprehensive classification systems in an attempt to cover as many different angles as possible. The OmniClass classification system introduced in Chapter III is an example of this, along with COBie and other classification standards closely related with IFC. A preliminary analysis of OmniClass is offered in Appendix (A), where it is showed how the extensional approach adopted leads to a number of redundancies and inconsistencies about the meaning of different entities, especially when multiple conflicting views need to be provided.

At a more fundamental level this is a problem of ontology, that results from the specific ontological commitments implicit in the conceptualization of IFC and similar data models and classification systems. In particular, the lack of a formal category for the description of processes, events and other time-dependent phenomena, restricts the degree of expressivity required to properly characterize design entities by virtue of what they *do*, and not by their structural configuration, constitution or shape.

Despite of these limitations, these standards are very relevant, making the important contribution of consolidating domain knowledge and making available in a standardized format to different members of the AEC industry. They also provide a foundation for future developments in automation and interoperability.

Therefore, the criticism is made to illustrate the difficult nature of the problem, and to propose an alternative approach to complement, and not necessarily replace, what is already available and widely adopted. In this particular case, the intent is to propose a method to formally describe the various functional roles that can be played, often *anonymously*, by different individual entities, at different moments in the life of a building system, along with the set of conditions under which such functional roles could be played.

The notion of 'anonymous' functional participation is key in the proposed formulation, because it makes evident the limitations of the extensional approach in the functional characterization of entities. To explain this further, the case of kettle can be used as example. Indeed, the fact that water is usually boiled by a kettle, does not mean that it cannot be boiled by other means. Conversely, a kettle could arguably play other functional roles besides its nominal function of 'boiling' water. If design entities like a kettle are defined by

extension, according to a rigid taxonomy of functions, then boiling water is pretty much the only thing the kettle can do.

On the other hand, whatever idiosyncratic means are used to boil water, most likely the object used will not be named 'kettle'. On the same token, any additional functional role that the kettle may fulfill beyond its nominal function, will not change its main denomination either. Therefore, due to the idiosyncratic nature of the functional participation taking place, the characterization of a functional role should not be restricted to nominal artifacts, i.e. artifacts that have functions explicitly allocated to them by name. On the contrary, artifacts playing a functional role should be treated as anonymous. This would allow generality and extensibility in the classification of possible members.

This idea is reinforced by the fact that, besides idiosyncratic, a functional role is often outside the direct control of certain stakeholders, such as designers or end users. It is because of this, that such anonymous entities playing a functional role are called *daemons* in this research. The question about intentionality, as the main factor to differentiate between functional and unintended roles [76], is not really critical, since the main goal here is to identify the entities with participation in the failure, and not who intended the failure, since most of the times no one really does. A similar argument can be given about 'good' intended functions, especially in the context of the built environments, where it is difficult to track all stakeholders, with all their possible, and often overlapping intentions.

8.2.5 Bracing and condensation daemon patterns

A function having a 'daemon' allocated as anonymous participant is called a 'daemon function', and together they conform a 'daemon pattern'. In this subsection two examples are presented, to demonstrate how the notions of daemon, daemon function and daemon pattern are used to define two classes of design entities by intension, therefore supporting the type of dynamic classification that is required to make functional modeling operational in the domain of building design.

The first example of intensional definition of design elements based on the formal characterization of structural and behavioral constraints is provided in the second case study

7.2. In this example, two classes of design components are defined intensionally, where specific members are classified without explicit reference to their names, i.e. anonymously. These included the class of moment-resistant connections and the class of bracing devices.

The conditions for membership include a basic set of structural and behavioral constraints, including conditions for internal parts and sub-functions. The listing below shows the definition for the class of bracing devices, presented again for convenience. Informally, the definition states that to be a member of this class, a design entity has to have at least one moment resistant connection as internal component. This in turn is defined also by intension, covering the general class of brackets that have at least two fasteners as their components. Thus, an individual bracket with just one fastener (e.g. one bolt, rivet or screw) is classified by the reasoner as a pin-connection, and therefore cannot be moment-resistant.

Listing 8.4: Intensional definition of bracing as daemon pattern.

```
1 Class: sol:Bracing_device
2   EquivalentTo:
3       (fr:has_component min 1 sol:Moment_resistant_connection)
4        and (fr:s_connected_to min 2 (fr:s_adjacent_to some
5             sol:Mounting_base_top))
6   SubClassOf:
7       inverse (fr:_DF_artifact) value sol:_fd4.2_bracing
```

Notice that the definition can be used for classification both at class and instance level. In this way other classes could be subsumed under the general class of bracing devices, etc.

According to the conceptualization proposed, the class Bracing_device correspond to a class of anonymous entities, referred generically as daemons. The daemon function _fd4 .2_bracing is indicated by the **SubClassOf** at the bottom of the listing. Its definition is presented below, where it can be seen that it approximate the definition $DevFunc(a, b_0, b_1)$ for a device-centric function in DOLCE-FR.

The main differences however, which remain consistent with the intended meaning, are that in this case two input behaviors b_0 are specified, while no explicit nominal artifact is given. Such anonymous artifacts, or 'bracing daemons' will be automatically inferred by

the reasoner, whenever the specified constraints are satisfied.

Listing 8.5: Assertion of bracing daemon function with anonymous artifact.

```
1 Individual: sol:_fd4.2_bracing
2     Facts:
3         fr:b0  sol:b_loading_seismic,
4         fr:b0  sol:b_loading_wind,
5         fr:b1  sol:b_bracing_capability
```

When the device-centric specification of a bracing function is allocated or bound to appropriate 'daemon' entities, the combination denotes a *daemon pattern*. In other words, a daemon pattern is a functional pattern, composed of a set of anonymous design entities bound to a function because an functional role in such function has been inferred. Such entities are said to be anonymous because their functional role cannot be inferred according to their main type or nominally allocated function. Defining this class of entities by intension allows automatic classification whenever the conditions of membership apply, both at the class and individual level. Given that such conditions are context-dependent, and the context during the design process is variable, the set of members for the class is also variable.

A final reason considered in this conceptualization, is that the terms *function, functional role* and *functional pattern* usually have a positive connotation. A function is something that is normally intended by someone, and therefore anything that contributes to such function is assumed to be good. The use of the term 'daemon' is intended to prevent that assumption. The definition by intension of the class of bracing devices, developed during the second case study demonstrates how a wind deflector, or any other artifact meeting the criteria, could also be used to fulfill the bracing function, which was required in the context of a ballasted PV racking system.

In a different behavior environment however, under a different set of requirements, bracing would be not only irrelevant, but actually detrimental in relation to what is needed. Therefore, there is no intrinsic value that makes this function positive or better than any other in principle. To demonstrate this idea, the concept of daemon patterns is used next

to represent a function that is not considered 'good' in general, i.e. a failure mode related to condensation in buildings. This definition will then be used to show how the proposed framework could be used to elucidate the aspect system of a failure mode, in the same manner that a normal function. During this process, other theoretical and implementation aspects of the research will be discussed.

The following listing provides the definition of the class Daemon_of_condensation, as it relates with any physical entity, artifact or otherwise, that is capable of releasing moisture into the air, thus having the potential to play a functional role in condensation as the function / failure of interest. These entities are first classified under the class of 'Humidifiers'. Then the actual participation of a humidifier in condensation is assessed by constraints specified in the daemon class. In this case, the only constraint is that a humidifier needs to be in direct contact with indoor air, i.e. s_directly_connected_to value a1.4_indoor_air.

Listing 8.6: Definition of a daemon entity for condensation as failure.

```
1 Class: cow:Daemon_of_condensation
2     EquivalentTo:
3         cow:Humidifiers
4          and (fr: s_directly_connected_to value cow:a1.4_indoor_air)
5     SubClassOf:
6        fr:Daemon
7          inverse (fr: _DF_artifact) value cow:fd_humidifier_daemon_function
```

Thus, the class Daemon is the equivalent to all humidifiers that are directly connected with indoor air. Inferred members of the class are then assigned as artifacts of the device (failure) function fd_humidifier_daemon_function. As mentioned earlier, a device function needs to be satisfiable in order to be processed by the reasoner.

The same principle applies to a failure defined in terms of a function. For that purpose, the class Satisfiable_daemon_function is defined, as specialization of satisfiable device functions. The definition in OWL-DL is presented in the next listing.

Listing 8.7: Definition for the class of satisfiable daemon functions.

```
Class: fr:Satisfiable_daemon_function
    EquivalentTo:
        fr:_DF_artifact some fr:Daemon
    SubClassOf:
        fr: Satisfiable_D_function
```

The formulation above allows the daemon function associated with humidifiers to be specified at the instance level without direct reference to any particular nominal artifact. This approach allows any set of artifacts responsible for the release of moisture to be inferred dynamically, according to different conditions of the behavior environment. Figure 8.2.5 shows the specification of the daemon function for humidifiers, where only the input and output behaviors b_0 and b_1 are explicitly asserted, on the bottom right of the figure.

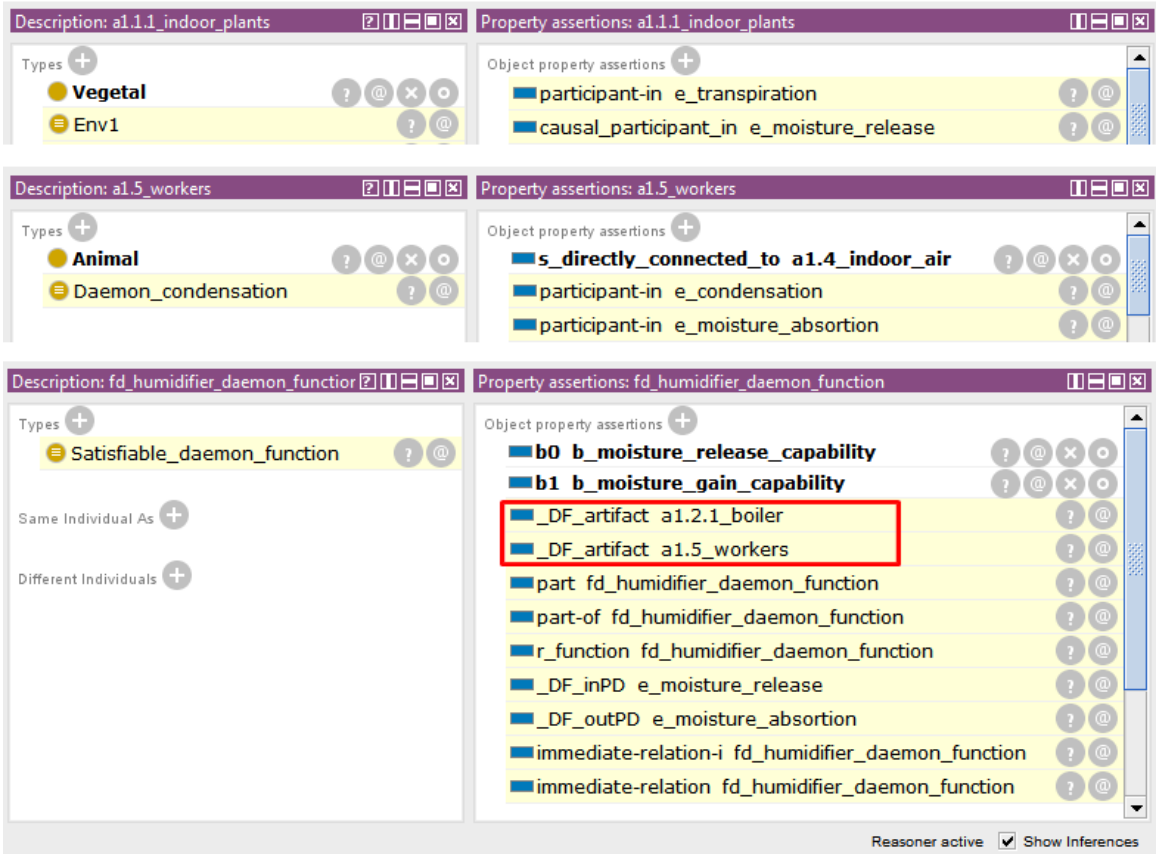

Figure 8.4: Inferred 'daemon' artifacts of satisfiable daemon function for condensation.

In the figure, it can be seen that any entity classified as 'humidifier' is classified as daemon, as long as such entity is connected with indoor air (bottom of the figure). Indoor plants are not classified as daemons even though they part of the indoor space and are classified as humidifiers, because they are not known by the reasoner to be in contact with indoor air, due to a lack of an asserted (or inferred) connection relation (top of the figure). Office workers in turn are both part of the space and are known to be in contact with indoor air (center of the figure).

8.2.6 Inference of failure based on daemon patterns

The *analogy* with the concept of geniuses and daemons is pertinent because both functions and failures often occur at the environment level by an unique combination of interaction patterns that are specific to a context, and under specific modes of deployment. Often these combinations are also outside the direct control of users. Therefore, it is only possible to have a more complete assessment of the nature of these interactions and their external effects, when such a context of use is provided at a higher level of abstraction. To do so, the specification of a function/failure from a high level perspective needs to include the specification of a sub-function/sub-failure specification from a lower level.

In Aspecto-FR, this can done by the nesting a functions/failures within others, thus supporting a recursive structure. The need for supporting this modeling feature was first discussed in Chapter VI (6.3.8), adopting a schema inspired by the SBF model developed by Goel et al. [157]. A detailed demonstration is provided in Chapter VII, particularly in the second and third case studies (in 7.2 and 7.3).

The listing below allows to exemplify this modeling feature in the context of the failure mode from an environment perspective, using an internal daemon function as input behavior. More specifically, input behavior of the e environment-centric 'failure' function f_failure_mode_condensation is a pointer to the output behavior b1 of the lower-level daemon function fd_humidifier_daemon_function, not showed in the listing. The immediate effect at the environment level, specified by the output b1 is condensation, constrained by the behavior b_condensation_capability.

Listing 8.8: Failure mode as environment-centric function with nested daemon function.

```
Individual: cow:f_failure_mode_condensation
    Types:
     fr:_E_function
    Facts:
     fr:_EF_in_environment kb:Env1,
     fr:_EF_nominal_artifact   cow:a1.1 _office_space ,
     fr:b0   cow:fd_humidifier_daemon_function,      // Pointer to internal subfunction/subfailure
     fr:b1   cow:b_condensation_capability
```

Following the terminology of SBF, the schema above could be read informally as "condensation occurs in the office space by the behavior of a humidifier daemon", where the behavior is indicated by b0, similarly to the use of the clause BY_BEHAVIOR as a pointer in SBF.

8.2.7 Aspect system of a failure

While the entities baring main responsibility in the production of condensation can already be identified by the reasoner, they are not the only participants that need to be considered for more complete assessment of the failure. For that purpose, the specification of an aspect system can be applied in the same way that it is done with an intended environment function.

Listing 8.9: Aspect system of condensation failure mode, in behavior environment Env1.

```
Class: cow:Aspect_system_condensation_failure
    EquivalentTo:
        (dc:part some (dc:participant-in some (fr:_c_causes some
                (inverse (fr:_DF_outPD) value cow:fe_failure_mode_condensation))))
         and (dc:proper-part-of value kb:Env1)
```

In any case, the condensation phenomena by itself might not be considered high-level enough as to demonstrate the validity of the proposed approach to support vertical integration of low and high level functions, at least in the context of Building Design. For that

purpose, it needs to be related with environment functions at a higher level of abstraction. In the areas of Architecture and Building Design, these usually correspond to programmatic requirements and functional performance mandates associated with the overall well-being of various users and occupants. It also includes other high level goals from relevant stakeholders, especially in the case of institutional and corporation buildings. The following subsection will address briefly this point, in order to demonstrate the theory behind this proof-of-concept.

8.2.8 Vertical integration based on functional roles

Regarding the high level functional mandates, literature in the subject has identified eight major Indoor Environmental Quality (IEQ) factors, that are especially relevant for the working environment.

While an in-depth treatment of the topic is certainly beyond the scope of this research, two are worth mentioning, including Indoor Air Quality (IAQ), and Biophilia. The former is a complex concept, composed of various time dependent physical and chemical processes, including relative humidity, temperature and level of air contaminants, which altogether may have significant negative effects in the well-being and productivity of occupants. The second factor, biophilia, is related to the positive psychological effects of being close to greenery, which is also concomitant with natural day lighting [4].

Within the limits of this short example, these two IEQ factors are treated as high-level soft functions, for which no detailed account of causality can be easily provided. Indeed, even more conventional mandates such as thermal comfort are difficult to define and measure, in part because the exact causal mechanisms are not totally understood. Yet, at the end of the day, these are some of the most important performance requirements of any building, or any part of the built environment to be occupied by humans.

In this context, the notion of contributive causes is useful, because it allows to describe causal links at very general level, so that expressions such as 'indoor plants contribute to the improvement of productivity' can be meaningful in design situations, while the more specific, low-level working principles might not be fully understood yet [224]. The additional

advantage is that a more precise, detailed account of causality would require making explicit all direct necessary causes, which could be computationally expensive.

The relationship between some intermediate level function and productivity has been mentioned in the presentation of the second case study, regarding the design of a flat roof PV racking system (in 7.2). In that case study, one of the goals of the design was to streamline the process of installation, by having a series of ergonomic features. However, the exact link between these ergonomic features and processes with productivity of installation was not made explicit.

In this case however, it would be necessary to establish such connection. While the assertion that plants contribute to productivity in the work environment seems acceptable, the assertion that condensation contributes to productivity is clearly less so. What can be said instead is that Indoor Air Quality (IAQ) is a complex state (`ST`) composed of several other perdurants, i.e. a mereological fusion, some of which have condensation as their contributive cause.

Indeed, it is known that excess of condensation can cause growth mold in buildings, and this process in turn can affect IAQ, due to spores released by the mold. Therefore, a causal chain can be established linking temperature, moisture, condensation, mold growth, IAQ, health problems, and productivity. Admitting however that describing this entire vertical relationship by means of causal relations might be problematic, even under the broad notion of contributive causes, a second approach has been developed, based on the notion of *function as a role*. With this, it is possible to reconcile different views on how condensation relates with IAQ, including causality, contributive or otherwise, and composition. Thus, it is possible to say that condensation contributes to mold growth, which is a proper part of the perdurant fusion called IAQ.

This is done in Aspecto-FR with the a OWL-DL property chain axiom, which integrates contributive cause and proper part relations. Among the theoretical advantages of this method, is that is supports vertical integration of functional descriptions that might be very different in type and meaning, while avoiding proliferation of multiple causal relations. It also adds a higher level of specificity, given the fact that functional roles are played in

relation to *output perdurants* only, as opposed to any given intermediate causal link. This restriction is defined by the rolification of the class of output perdurants, denoted by r_outpd.

Listing 8.10: Functional role relation, based on causal contribution and composition.

```
ObjectProperty: fr:functional_role_in
    SubPropertyChain:
        fr:_c_causes o proper−part−of o fr:r_outpd,
        dc:proper−part−of o fr:r_outpd
```

Now it is possible to reformulate the definition of aspect system given for condensation as failure mode, but this time from a higher level of abstraction, considering the functional viewpoint of productivity. The listing below presents this new definition, which adds the relation functional_role_in immediately after the contributive cause relation.

Listing 8.11: Aspect system of productivity, with functional roles and contributive causes.

```
(dc:part some (dc:participant−in some (fr:_c_causes some
            (fr:functional_role_in some
                (inverse (fr:_EF_outPD) value cow:fe_productivity)))))
and (dc:proper−part−of value kb:Env1)
```

Figure 8.2.8 shows a comparison for the aspect systems of two different functions. First on the left, the aspect system for the indoor air quality. On the right, the complete aspect system for productivity, based on the previous axiom.

The DL query expressions do not inform however if the failure mode for condensation is actually 'in place', in terms of having an impact in IAQ and productivity. While there is an aspect system inferred, it is not clear which members are simple co-participants, and which are the main participant 'daemons', if there are any.

Under the constraints specified for the daemon function, either boiler and indoor plants can be part of the office space, without having any main participation in the condensation problem. To clarify this point, according to the criteria of the research hypothesis, a final SPARQL query is made next.

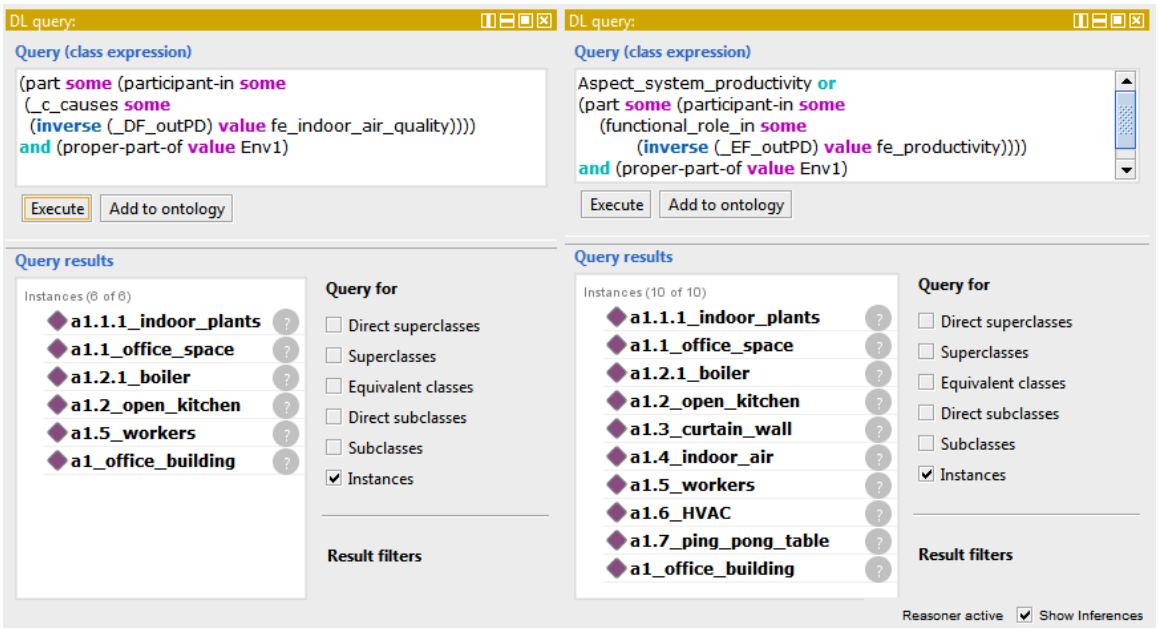

Figure 8.5: Comparison of three aspect systems related to vertical integration of functions.

Listing 8.12: SPARQL query on aspect system of productivity and possible failure daemons.

```
SELECT (?a AS ?aspect)(?t AS ?daemon)(?p AS ?participation)
      (?f AS ?function)(?b AS ?by_behavior)
WHERE { ?a a cow:Aspect_system_productivity_full
      OPTIONAL{?a a cow:Daemon .
      ?a fr:_b_has_capability ?b .
      ?b fr:behavior_constraint_on_perdurant ?p .
      ?c fr:_c_caused_by ?p ;
         a fr:OutPD ;
         fr:functional_role_in cow:e_productivity .
      ?f fr:_EF_outPD ?c .}
      OPTIONAL{?a a ?o.?o rdfs:label ?t}
FILTER(?a != cow:a1.5_workers) }
ORDER by desc(?a)
```

In the query, any daemon involved in downplaying the achievement of productivity is returned. In the model, only daemons associated with the failure mode of condensation have been specified. Therefore these are the only ones in the list, along with other participants in the aspect system of productivity. However, identified daemons have their problematic type

of participation identified, along with the behavioral constraints or capabilities associated. In this case, the boiler and indoor plants are the 'culprits', by being the main participants in the failure mode. A short explanation is given by the variable `?by_behavior`, which denotes the main behavioral capability responsible for the participation. This can be seen in Figure 8.2.8, along with the rest of returned results.

Office workers have been filtered out, since they are the end users of the space after all. Otherwise they would be also identified as 'daemons' of the failure as well. It is important to point out that the same approach could be used to identify the 'geniuses' of a function, that is, entities that contribute or play a positive functional role in the satisfaction of an environment-centric function, without having been asserted previously as 'nominal' or main participants of the function.

With this last query, the proposed approach for the vertical integration of functions is demonstrated, based on a reduced set of entity and relationship types. Moreover, the applicability of the definitions of device and environment functions, translated from DOLCE-FR into OWL-DL, provides an expressive conceptualization to support a formal implementation of the notion of aspect systems. The case studies developed, including this last one, have introduced the most relevant features and main rationale behind the implementation of a proof-of-concept, suggesting a promising approach for the development of future functional modeling applications for Building Design.

At the more specific level, the case studies allow the research hypothesis to be verified, based on the satisfaction of the two main criteria laid out in Chapter V, namely, the criteria of multi-functionality and functional integration. The implementation of the proof-of-concept, along with the series of queries developed for each case study scenario demonstrate that both dimensions can be described based on this prototype. In particular, the goal of supporting the elucidation of aspect systems, as functionally-driven cross-cutting aggregation, has been achieved, according to the objectives and scope defined for this research.

Snap SPARQL Query:

```
SELECT  (?a AS ?aspect)(?t AS ?daemon)(?p AS ?participation)(?f AS ?function) (?b AS ?by_behavior)
WHERE {
            ?a a cow:Aspect_system_productivity_full .
            OPTIONAL{?a a cow:Daemon .
            ?a fr:_b_has_capability ?b .
            ?b fr:behavior_constraint_on_perdurant ?p .
            ?c fr:_c_caused_by ?p ;
                        a fr:OutPD ;
                        fr:functional_role_in cow:e_productivity .
            ?f fr:_EF_outPD ?c .}
            OPTIONAL{?a a ?o.?o rdfs:label ?t}
            FILTER(?a != cow:a1.5_workers) }
ORDER by desc(?a)
```

Execute

?aspect	?daemon	?participation	?function	?by_behavior
cow:a1_office_building				
cow:a1.7_ping_pong_table				
cow:a1.6_HVAC				
cow:a1.4_indoor_air				
cow:a1.3_curtain_wall	Condensable_surface			
cow:a1.2_open_kitchen				
cow:a1.2.1_boiler	Boiler	cow:e_moisture_absortion	cow:fe_failure_mode_condensation	cow:b_moisture_exchange_capability
cow:a1.2.1_boiler	Boiler	cow:e_moisture_absortion	cow:fe_indoor_air_quality	cow:b_moisture_exchange_capability
cow:a1.1_office_space				
cow:a1.1.1_indoor_plants	Vegetal	cow:e_moisture_absortion	cow:fe_failure_mode_condensation	cow:b_moisture_exchange_capability
cow:a1.1.1_indoor_plants	Vegetal	cow:e_transpiration	cow:fe_failure_mode_condensation	cow:b_by_photosynthesis
cow:a1.1.1_indoor_plants	Vegetal	cow:e_moisture_absortion	cow:fe_indoor_air_quality	cow:b_moisture_exchange_capability
cow:a1.1.1_indoor_plants	Vegetal	cow:e_transpiration	cow:fe_indoor_air_quality	cow:b_by_photosynthesis

13 results

Figure 8.6: SPARQL query on aspect system for productivity with associated failure daemons.

8.3 Contributions, limitations and future work

In this section a summary of the main contributions of this research is provided, from both a theoretical and implementation perspectives. The theoretical perspective includes aspects of computation, knowledge representation and ontology as they apply to design in general, and Architecture and Building Design in specific. From the implementation perspective, only most relevant aspects related with the proof-of-concept are included. This followed by the main limitations identified, and recommendations for future work. Some final remarks are offered at the end.

8.3.1 General research problem

- The research identifies the general requirements for the implementation of an operationally robust functional modeling framework for the domains of Architecture Engineering and Construction (AEC).

- The identification of these requirements is based on a comprehensive literature review, including the history and evolution of computer-aided design, and building information models. The review also included relevant work in the topic of functional representation, particularly Artificial Intelligence, Engineering Design, Systems Engineering and Applied Ontology.

- Among these general modeling requirements, a more specific subset has been identified to support interdisciplinary building systems integration and performance-based design.

- Within this context, the research proposes the formalization of the concept of Aspect Systems, as fundamental abstraction required to describe the set of building entities from different building subsystems and levels of abstraction that have the potential to play a functional role in the satisfaction of multiple performance requirements.

- The research proposes an operational formalization of aspect systems, grounded on a formal, model-based representation of design functions, intended to support a variety of design tasks, including:

- Formal, model-based specification of functions and performance requirements.
- Search and selection of technical models and components from BIM libraries.
- Iterative cycles of alternative generation and design-analysis integration.
- Multi-criteria performance evaluation, trade-off analysis and decision-making.

8.3.2 Specific problem and scope of the research

- The research specifically addresses the problem of providing a formal characterization of building functions and associated concepts, required to support the formalization of the concept of aspect systems. Altogether, the integrated formalization of function, behavior and aspect system should provide the basis for a operationally robust functional modeling framework. Among the main requirements for such framework is the ability to enable multiple functional views of a building model. These functional viewpoints are generalized in this research as follows:
  - **Supply and demand:** A common representation of functions that serve both, the specification of required functions, before or during design and construction (demand side), and the specification of functions provided, after design, fabrication or construction (supply side).
  - **Multiple levels of abstraction:** Inference of aspect systems require the identification of functional interactions across different sub-systems and levels of abstraction. This implies the need for a general, compatible representation for the different, domain-specific meanings of function used in building design. Such compatibility has to support integration in two orthogonal directions.
    - **Horizontal integration (domain-specific functional meaning):** Integration of domain-specific meaning of functions associated to different technical sub-systems at the same level of aggregation (e.g. electrical and mechanical subsystems, etc.).

    - **Vertical integration (hard and soft functions):** Integration of functional meaning across different levels of abstraction. This involves the problem of how the semantic content of low-level functions, usually technical and quantifiable (e.g. heat pumps), relate with the semantics of intermediate and high-level functions, which usually are not technical or easily quantifiable (e.g. productivity).

- **Distal functional dependencies:** Two additional functional perspectives required have to deal with how entities that are distant from each other in space or time are described as functionally related. In the first case, a structural relation needs to exist, but for which there is no explicit description in the model. This problem relates with the topology of physical connections, which in general are only described if the nature of functional interactions are known beforehand. The second case relates with the functional implications of past events in the state of affairs of the present. However, during design, the relevance of this type of relations stems from situations where design decisions made to improve performance aspects of a specific life-cycle requirement, have unanticipated consequences in early or late life-cycle requirements.

- **Nominal and anonymous functional roles:** This class of functional viewpoints relate to the fact that many parts of buildings perform more than one main nominal function. At the same time, many functions are affected by the participation of multiple parts, either negatively or positively. The set of these parts is the aspect system of the function, and the specific goal of this research is precisely to enable the identification of these parts. However, not all elements in a BIM model, if any, contain explicit descriptions of their main nominal function, let alone additional, and often idiosyncratic functions. Any additional functional role performed, intended by design or accidental, is either implicit in the model or unknown. Therefore, the identification of the members of an aspect system requires the explicit characterization of these additional functional roles. For that

purpose, the formalization of various functional and behavioral aspects of building elements is required, so that all relevant functional roles can be identified by means of inference.

Current BIM applications are incapable of supporting the type of functional views, especially if they require automatic classification and reasoning. Among the main reasons, the following theoretical problems are worth mentioning:

- **Lack of formalization:** Building information models lack formal characterization of functional and behavioral aspects associated with buildings.
- **Partial ontological commitment:** The lack of formalization is in part result of ontological commitments implicit in the conceptualization of building information models in general. This limitation restricts the set of ontological categories and relations available to formally describe phenomena of interest, particularly time-dependent phenomena that need to be captured by different functional viewpoints.
- **Reliance on definitions by extension:** This problem was analyzed in depth in Chapter III dedicated to the analysis of IFC. The ontological and practical implications of extensional definitions were discussed, especially in relation to the lack of consistent criteria for the classification of design entities based on property sets.

To address both theoretical limitations, the research adopts the formalization of the Functional Representation schema under the DOLCE foundational ontology, which together provides a sound ontological framework to formally represent processes, events, states and other occurrences that need to be described by means of proper behavioral and functional abstractions required in design.

By recognizing of the general category of time-dependent phenomena, i.e. perdurants, a series of useful abstractions are provided, which constitute the basis for the formulation of the research hypothesis, and the proof-of-concept implementation used as method of validation.

8.3.3 Hypothesis and methodology

The hypothesis of the research is that a subset of axioms given by the formalization of the Functional Representation schema under DOLCE, called in this research DOLCE-FR, provides the basic vocabulary required to formally describe the functional viewpoints identified in as the specific research problem, as requirements for the formalization of the concept of aspect systems. At a more specific level, the hypothesis is that the translation of this subset of axioms, from First-Order Logic into the Description Logic version of the Web Ontology language, enables the operational formalization of aspect systems, by taking advantage of automatic inference capabilities of available OWL-DL reasoners.

Regarding the key functional concepts translated from DOLCE-FR, and that constitute the theoretical core of the implementation, is the dual meaning of function, originally proposed by Chandrasekaran and Josephson. These include:

- Function as behavior constraint
- Function as intentional role

Regarding the research methodology for hypothesis validation, this involved the development of a proof-of-concept functional modeling framework, called Aspecto-FR. This framework was used in the development of four different case studies. Three of these case studies were presented in Chapter VII, and the fourth one was introduced previously in this chapter as part of the final discussion.

The main aspects of the implementation of Aspecto-FR were presented in Chapter VI, with some more specific axioms and definitions introduced during the presentation of individual case studies. An important criteria in the formulation of these axioms was the definition of entities by *intension*, based on conjunction of relevant properties and constraints. This criteria was applied for the characterization of several devices and functions, addressing in this way the problem associated with extensional definitions identified as one of the main theoretical limitations of current building information models (e.g. IFC).

Finally, some new relations were also defined. This was done using OWL object properties, with different restrictions and characteristics applied (e.g. cardinality, transitivity,

etc.). Property chains were also used, often in conjunction with the method of class rolification. These object properties, along with a few auxiliary SWRL rules implemented to support inferences required, are explained in detail in Chapter VII and VIII. The complete OWL-DL definitions are included in the listings of appendixes.

Regarding the criteria for hypothesis validation, this was formulated based on the ability of the framework implementation to support inferences required over the models developed for the case studies. The assessment of results was made by means various queries reflecting the functional viewpoints discussed.

8.3.4 Contribution

The main contributions of this research can be summarized as follows:

1. A functional modeling framework for building design has been formulated, with the goal of providing computational support in building systems integration and performance based-design.

2. The research develops a common formalism for the representation of different building functions, capable of supporting multiple functional viewpoints, and across multiple levels of abstraction. In particular, the research proposes and demonstrates an approach for the formal characterization of the concept of aspect system, as cross-cutting aggregation of different building parts considered relevant from an arbitrary functional perspective.

3. A proof-of-concept has been implemented in Description Logic (OWL-DL) based on the translation of a subset of the axioms from the formalization of the Functional Representation schema (FR), under the DOLCE foundational ontology (DOLCE-FR).

4. The research demonstrates how the implementation of a small vocabulary of concepts translated from DOLCE-FR allows the specification of arbitrary functional viewpoints, and supports the characterization of associated aspect systems based on inferences over two main entity types (i.e. DOLCE categories) and three main relation types. These main abstractions implemented in the research are:

- participation relation of endurants in perdurants.
- causal relation between perdurants.
- composition relations among endurants or among perdurants.

5. The research demonstrates how more complex abstractions can be built on top of this basic formal vocabulary. In particular, it demonstrates an approach for the implementation in OWL-DL of the notions of device-centric and environment-centric function formalized in DOLCE-FR.

6. The research provides a series of examples and case studies, demonstrating how these two notions of function, along with three main types of relations could support the inference of functional interactions across different levels of abstraction, based on the general meaning of *function as a role*.

7. In particular, the research proposes an approach for intensional characterization of functional entities, based on the concept of functional (daemon) patterns. The research demonstrates through a series of examples how this method can be applied to support inference of additional functional roles played by artifacts within a context, by means of specification of constraints.

8. Regarding the generalization of the proposed model, is showed how the implemented definitions of device and environment functions, functional (daemon) patterns, and aspect systems could be used to represent failure modes, and the inference of elements having major participation in them.

9. A comprehensive description of the implementation is offered, along with theoretical aspects involved. Four case studies are developed, as part of the methodology for testing and validation of the research hypothesis, based on a series of queries reflecting different functional viewpoints. Results are presented, limitations identified and recommendations for future work are outlined.

8.4 Limitations and future work

The main limitations identified in the research can be classified under three main theoretical dimensions: ontological, disciplinary and computational. While the list is not exhaustive, the limitations selected indicate some of the most relevant steps for future research, which are included in the next subsections.

8.4.1 Ontological limitations and future work

- **Causal relations:** The approach adopted in this research for the definition of causal relations tried to reach a balance between proper formalization from an ontological perspective, and practical considerations related with implementation, as well as the most commonsense use of the notion of causality in design practice. Arguably, the complexity of the topic is beyond the scope of this research, and more work will be needed to provide a better ontological characterization of causal relations. In particular a categorization of constraints associated with various causal relations is required, including the specific types of perdurants and participant endurants involved. The lack of such criteria suggests the risk of combinatorial explosion of 'causes', that would make difficult to obtain the meaningful inferences, especially in larger models. For this reason, additional constraints over more specific types of causal relations need to defined in the future.

- **Categorization of engineering perdurants:** Another limitation identified, particularly during the development of the knowledge base models for the case studies, was the lack of a consistent criteria to categorize the perdurants associated to different participation relations and functions. This limitation is closely related with the previous problem regarding categorization of causal relations. In general, the decision was made that output perdurants of functions would be defined under the DOLCE category of state (`ST`), which seemed appropriate for the description of behavioral properties in terms of discrete, atomic time units, akin to the specification of performance indicators related to the concept of aspect systems. However it is not clear if this is necessarily the only option, based in part on the type of causal relation and

participants involved. Moreover, states as complex aggregations of other perdurants (i.e. mereological fusions), and the role or *proper-part* relations will need to be further investigated. Moreover, a more complete description of the relationship among composition, participation, and causality indicate the need for a formal classification of behaviors and behavior constraints that are relevant in design.

- **Time and temporal relations:** Another limitation is the lack of temporal indexation in the specification of participation relations. This is because the primary goal was to keep the vocabulary of entities and relations as simple as possible, to reduce complexity in the definitions, but also to minimize overhead associated modeling and reasoning. The problem however, is that potentially relevant time-dependent relations cannot be not made explicit, forcing the adoption of assumptions that may be opaque or invalid from a domain perspective. Moreover, to meet the prospect of integration with design and analysis applications, especially to support the development of simulation models, the formal treatment of temporal relations needs to be addressed in the long term.

8.4.2 Computational limitations and future work

- **Types of inference:** The proof-of-concept developed in this research relies on theorem proving based on Description Logic (DL) semantics, or more specifically the OWL version of DL. This provides the advantage of the availability of 'off-the-shelf' reasoners, such as Pellet, to perform the inferences required, assuming conditions of *decidability*. Clearly, this imposes restrictions on what can be represented in the model and how, as well as the usefulness of inferences provided, especially under real-world scenarios. Therefore, given the multi-dimension nature of design, and the diversity of functional perspectives required, a toolkit of additional methods of representation and reasoning methods will most likely to be needed as part of a practical, and operationally robust, functional modeling application. Next steps should consider specific real-world scenarios where this need becomes evident, to identify the types of abstraction and reasoning required.

- **Reasoning performance:** The performance of DL reasoners depends fundamentally on the expressivity of the ontology. Other metrics however, such as the number of axioms defined for classes, properties and individual assertions also play a role. An overview of the different levels of expressivity and metrics associated with the main ontology model, as well as the different knowledge base models developed for the case studies, is presented in table 32 of Chapter VII. In most cases, the DL expressivity of the models was $\mathcal{SROIF}(\mathcal{D})$, which is considered in the class of reasoning problems with complexity `ExpTime-Hard`. The only exception was case study 3 (bundled together with case study 2 in the table), which has a DL expressivity $\mathcal{SROIQ}(\mathcal{D})$, with reasoning complexity considered `NExpTime-Hard` [334]. An explanation for the latter is the use of complex role inclusions in some object property axioms attempted during the development of the model. The impact in reasoning performance can be seen in the table, where running time far exceeded the other models, reaching up to three minutes in the worst scenario, where the rest stayed below one minute. Therefore, it is most certain that there is significant room for refinement and optimization, and future work needs to address that. Despite the effort to keep the prototype as lean as possible, some few auxiliary classes and properties exist, that might not be needed at all after more systematic evaluation. Additionally, some of the functionality of the model could be outsourced to external applications, reliving the reasoner from tasks that could be better handled externally.

- **Query performance:** Regarding queries made in SPARQL, performance in general decreased with the complexity of the semantic graph inferred from the asserted model. In this aspect, case study 3 had the worst performance for queries involving combination of property chains and transitive properties (e.g. causal_participation). In the worst scenario, query time took six minutes (326802ms), where for the last case study it took about less than 17 seconds (16537ms) for the same query expression. The impact of complex role inclusions discussed above will need to be studied.

8.4.3 Domain-specific limitations and future work

- **Integration with IFC / BIM:** Currently, the test-bed implementation relies on an internal, simplified logical representation of a design model. However, the use of the Web Ontology Language (OWL) as main formalism is intended to support future integration with BIM applications operating within the Semantic Web / Linked Data environment, and as part of a larger of ecosystem of federated models distributed over the web. A more detailed overview of the envisioned framework within the Semantic Web / Linked Data environment was presented in Chapter VI (6.2). In particular, it is envisioned that a specific form of integration with BIM models could be based on a recent OWL implementation of IFC [51], [250]. Immediate next steps involve exploring this possibility, by redeveloping some of the case studies with IFC data. This would allow a more realistic assessment of opportunities and roadblocks for practical applications, especially in relation to the information requirements and complexity of 'real-world' models and design workflows, including the integration with analysis and simulation applications. In this regard, the following points are made:

  - **Automatic extraction of model views and conceptual alignment:** The proposed functional modeling framework relies on dynamic classification and mapping of functional relationships based on a combination of rules and logical inferences. In an ideal world, this capability would be enough to generate the necessary views from BIM models to support different tasks related with functional modeling. Most likely however, is the scenario where certain conceptual alignment would be needed, based on some post-processing of BIM data by the source application or some intermediate layer. In any case, it is not anticipated that a major overhaul in the internal conceptualization or definitions of building models, such as IFC would be required. Therefore, the work presented does not suggest the need for 'all or nothing' approach for integration. Instead, a more tactical approach is envisioned, based on incremental efforts of conceptual alignment among models.

– **Characterization of functional roles based on intensional definitions:** Some examples of how model views can be extracted from IFC models based on SWRL rules are provided in [125]. This mechanism illustrates a possible method towards conceptual alignment with the semantics of other ontologies and data models. For instance, once a model view is extracted, other rules can be applied for semantic characterization under a different perspective. In the case of IFC, most of the information provided refers to structural entities and relationships. Therefore, assuming the appropriate mapping rules, such views could be 'qualified' by behavioral constraints, according to the conceptualization proposed in DOLCE-FR / Aspecto-FR. This in turn would enable the classification of extracted IFC entities based on functional meaning defined by intension in the ontology model. Such intensional definitions can be composed as reusable modular patterns, embedding different types of functional knowledge relevant in building design. Examples of this are provided in case studies 2 and 3, particularly in the definition of moment-resistant brackets and bracing devices (in 7.2 and 7.3), as well in case study 4 introduced in this chapter, addressing failure patterns (8.2).

– **Integration of functional, behavioral and structural abstractions:** While it is clear that the structural models used for the development of the case studies are rather simplistic, it is also clear that a functional modeling application with reasoning capabilities will not need high fidelity models involving topological and geometric information either. At least not in the short term. What is needed are structural representations at an intermediate level of abstraction, grounded on a common, ontology-driven formalization, so that functional modeling and reasoning applications could be more seamlessly integrated without significant loss of structural meaning. Currently, approaches based on the extraction of model subsets, or derivation of model views (e.g. IFC Model View Definitions) lack the level of formalization required to support functional modeling and reasoning, for various reasons discussed in Chapter III (3).

– **Functional models for design-analysis integration:** Similar problems related to the lack formalization arise regarding integration with engineering analysis and simulation applications. Unfortunately, many of these applications are built on top of very narrow, domain-specific, and often idiosyncratic conceptualizations, and without much transparency regarding the criteria adopted in the definition of entities, relations and properties. This is problematic not only because it hinders the prospects of integration with functional modeling and reasoning, but also because valuable domain knowledge is hidden behind the source code of such applications, becoming inaccessible for verification, refinement and reuse. To address this problem, a formal characterization of the different entities that are relevant in building design needs to be adopted, grounded on a rigorous ontological framework that is shared among different domain applications.

  Among the different design activities requiring better computational support, and for which such characterization would provide a valuable contribution, the specification and management of functional requirements stands out. A formal, model-based representation of functions, grounded on an ontology of time-dependent phenomena, i.e. perdurants or occurrents, and enhanced with sound reasoning capabilities would be a first, necessary step to enable not only a more systematic formulation of functional goals and performance requirements, but also a more flexible framework for managing different design-analysis scenarios and workflows. In particular, it would provide the computational infrastructure required to enable the diversity of functional viewpoints that drive the use of performance analysis and simulation applications. Therefore, future work will need to solve the problem of providing a model-based representation of functions to describe both the demand and supply side in similar terms, so that integration of design and analysis applications could be driven by a common formalism.

8.5 Final remarks

The mention to the concept of *genius loci*, which eventually led to the adoption of the term 'daemon', to denote a specific type of functional abstraction developed in this work, was not intended to be just an anecdote, or a shallow analogy. Quite the contrary. This concept has always played a fundamental role in Architecture, taking different names in different cultures throughout history. The work of theoretician Christian Norberg-Schulz in the phenomenology of daily environments, contributed to the revival of the idea during the 1980s. The underlying principle is the connectedness that exists among the multiple levels of reality, as it occurs within a particular environment, and which gives the unique character of a place. This line of thought is key in the work of Christopher Alexander, especially in his work on Pattern Languages, which has an important influence in many disciplines beyond Architecture. The work of Louis Kahn, John Habraken, and other architects of the structuralist school share a similar motivation.

The work presented here is conceptually aligned with this set of principles and motivations. Above all, it shares the ideal that the design of the built environment should be at the service of people, and the diversity of values, experiences and perspectives that should co-exist in society. To that end, the multidimensional nature of reality needs to be recognized and given priority over other design considerations, which are often biased by very superficial reasons.

The main difference with that line of work however, is that this research does not attempt to provide any method of design, that could lead to a 'synthesis of the form' or the generation geometric shapes. The main goal is to expose the hidden structure of relationships that need to be taken in consideration from multiple perspectives during the design process.

Due to the increasing complexity of the problems faced by society, this challenge cannot be handled by any single discipline. For this reason, even though the fundamental motivation for this research is deeply grounded in the field of Architecture, it is intended to be a contribution for problems that are intrinsically interdisciplinary.

Architecture used to be - still is - a discipline concerned with the problem of connecting multiple levels of reality. Unfortunately, the urge for novelty, especially in terms of geometric

form, and style, has somewhat weakened that concern, slowing the progress of theories, and the development of tools and methods to address more fundamental problems facing society.

In a time where the constant production of new things seems to take precedence over the real purpose and value of such things, perhaps what is really needed is a new appreciation of what is already available around us, by making more evident the deeply interconnected structure of reality. The research presented in this dissertation attempts to make a contribution in that direction, by proposing a computational framework to represent explicitly some of these connections, as they occur within the built environment.

APPENDIX A

ANALYSIS OF OMNICLASS^{TM}

A.1 OmniClass^{TM} Table 41 - Materials

According to OmniClass^{TM} Table 41, materials are defined as "basic substances used in construction or to manufacture products and other items used in construction. These substances may be raw materials or refined compounds, and are claimed to be presented in the OmniClass^{TM} materials table *entirely without reference to their form.* For example, the description of Aluminum in terms of its chemical composition is in the scope of Table 41, whereas Aluminum in the form of bars, sheets or blocks belong to Table 23 - Products. Hence, the main criterion for differentiation between materials and products in OmniClass^{TM} is said to be form.

Yet, certain materials such as "sand" are described both in terms of their chemical composition (i.e. silicon dioxide) as well as their form. The argument made is that some materials are encountered in nature already in an useful, recognizable form as commodity. This is the case of sand, which is used both "as constituent material for other products as well as a material that can be used satnd-alone "in its natural form". For this reason sand is described in both OmniClass^{TM} Table 41- Materials and OmniClass^{TM} Table 23 - Products [82].

However, what makes sand different in each case is not its form, but the way it is used, either as part of another material (e.g. mortar), or as part of an assembly (e.g. cushion layer under a brick pavement). To complicate the issue even further, it can be argued that chemical compositions can also be characterized in terms of form, i.e. the structural configuration at the atomic and molecular level. Following this line of thought, the difference is not so much an issue of form as it is of scale. Products then are different because they are typically *formed* at a larger scale than the chemical compositions of materials. However this argument overlooks the development of manufacturing capabilities at the nano-scale.

At the end, OmniClass^{TM} attempts to avoid this problem of interpretation by using different definition approaches. Table 34 below provides a sample of these approaches. In some cases, the definition refers to how materials are made or their (geographic) origin. For example, see entries for Limestone, Porcelain and White Oak. In other cases, aspects of their chemical composition is provided (e.g. Granite). A third approach is the description of key behavioral properties (e.g. Porcelain, Asbestos, Carbon Fiber). Yet, a fourth approach is to describe typical applications of materials in construction, either as constituents of other materials and products (e.g. Aluminum, Brick, Portland Cement and Fly Ash).

A.2 OmniClass^{TM} Table 23 - Products

Products in the OmniClass^{TM} Table 23 are defined as "components or assemblies of components intended for permanent incorporation into construction entities." In other words, they are the building blocks used for construction, which may range from single manufacture items, to assemblies of multiple parts or manufactured stand-alone systems. However in the discussion section provided in Table 23, products are categorized *without regard of their application.* As example given, a product such as a glass panel may be used in a window, a cabinet shelving, etc. Because each of these uses are considered different work results, they are classified under Table 22 - Work Results.

However, similarly to the case of materials, the criterion of *use* or *application* does not seem consistent across product classifications and definitions. In general, the specific application is implicit in the titles (i.e names) of higher level categories of the table (e.g. Level 2 Title), or in the titles of products themselves. For instance, the category 23-11 25 00 - Site Barrier Products. which is defined as comprising "products that divide and / or protect a given site". Under this category certain products are described with a very specific application that is explicit in their names. The same applies for different types of showers, which may range from nursing, to patient specific showers (e.g. psychiatric, physically challenged), to emergency showers.

As can be seen from Table 35, the classification of products is inconsistent with the stated criterion of excluding product applications. In general, references to applications is either

Table 34: Appendix A: Sample of Materials. OmniClass Table 41

OmniClass™ Table 41 - Materials		
Number	**Title**	**Definitions**
41-10 10 60 11	Aluminum	Atomic Number 13; alloys are *used primarily in windows, doors, cladding.*
41-30 10 11 11	Granite	A naturally occurring type of coarse-grained igneous rock.
41-30 10 13 11	Limestone	A naturally occurring rock produced through layering and sedimentary collection of calcite and aragonite typically produced from skeletal fragments of marine life.
41-30 10 25 13 11	Brick	Typically bound with mortar, brick blocks are a common building product *used to construct nearly every type of vertical structure.*
41-30 10 25 13 19	Porcelain	Primarily formed from Kaolin, this material originated in China and is recognized by its white color. *Very resistant to thermal shock and considerably strong.*
41-30 10 25 19 11	Asbestos	A compound composed of six naturally occurring silicate minerals. *Historically used for its sound absorption, tensile strength and resistance to fire, heat and electrical damage, this substance has been found to be highly toxic* and is no longer as extensively used.
41-30 10 25 19 15 11	Portland Cement	The most common type of cement in use worldwide. *Typically used in concrete.*
41-30 10 27 11	Carbon Fiber	A material consisting of thin crystalline carbon filaments. *Typically used for its surprising strength to weight ratio.*
41-30 10 27 15	Fly Ash	Ash that rises with other gases from combustion. *Typically used in products such as Portland Cement, Roller-compacted Concrete* and *applications such as soil stabilization as well as waste treatment.*
41-30 30 11 19 13 11	White Oak	One of the most pre-eminent hardwoods of eastern North America.

in the title of the categories, in the name of individual entries or in their textual definitions. Similarly to materials in Table 41, products are also defined sometimes simultaneously in terms of structural composition, origin or form.

For example, ceiling panels are classified under the sub-category 23-15 13 00 "Multi-Function Interior Coverings, Claddings, Linings", while others under 23-15 19 00 "Ceiling Coverings, Claddings, and Linings". Some ceiling panel names in the first sub-category indicate the their material composition (e.g. Metal Ceiling Panels, Wood Ceiling Panels, etc.) whereas ceiling panel names in the second sub-category refer to either their form (e.g. Curved Ceiling Panels), some implicit behavioral property (e.g. Mirror Ceiling Panels which imply image reflectiveness), or a specific functional purpose (e.g. Acoustic Ceiling Panels). No further textual definition is provided for any of these, since the names themselves seem to suffice.

A.3 OmniClass^{TM} Table 22 - Works Results

As mentioned earlier, according to OmniClass$^{\text{TM}}$ discussion section of Table 23, the specific application of building products is supposed to be described in Table 22 - Work Results. Such work results are defined as the outcome of construction activities, or by subsequent alteration, maintenance or demolition processes. Work results typically involve one or more of the following conditions, necessary to support contracting, planning, scheduling and budgeting among other activities, such as:

- A particular skill or trade.
- Construction resources.
- A resulting construction entity.
- Temporary or other preparatory work.

By reviewing these conditions and analyzing the various entries in Table 22, it is possible to verify that the description of product applications, uses or purposes as mentioned in the discussion section of Table 23 is actually not part of its scope.

Table 35: Appendix A: Sample of Products. OmniClass Table 23

OmniClass™ Table 23 - Products		
Number	**Title**	**Definitions**
23-11 25 00	**Site Barrier Products**	Products which divide and or protect a given site.
23-11 25 15 15 11	Anti Ram Sliding Gates	None.
23-11 25 15 15 13	Anti Climb Sliding Gates	None.
23-13 13 00	**Binding Agents and Admixtures**	especially formulated products which modify the properties of either paint or concrete to give it certain characteristics not obtainable with plain mixes.
23-13 13 13 19	Cement Setting Retarders	None.
23-13 15 11 15	Low Density Concretes	None.
23-13 17 00	**Profiles**	Products or materials involved with supporting the structural and exterior enclosure.
23-13 17 15 11	Precast Hollow Core Sheets	Prefabricated concrete slabs used to support the structure, can be used in parking garages or apartment buildings.
23-13 17 15 15	Precast Double Tees	Two legs beneath a slab to span a long distance.
23-15 13 00	**Multi-Function Interior Coverings, Claddings, Linings**	Interior covering, cladding and lining products used inside the facility to finish surfaces and divide spaces that have more than one function.
23-15 13 21 11	Metal Ceiling Panels	None.
23-15 13 21 15	Wood Ceiling Panels	None.
23-15 19 00	**Ceiling Coverings, Claddings, and Linings**	Ceiling coverings, claddings, and linings used inside the facility to finish surfaces.
23-15 19 15 13 15	Curved Ceiling Panels	None.
23-15 19 15 13 13	Mirror Ceiling Panels	None.
23-15 19 23 11 13 13	Acoustical Ceiling Panels	None.

Instead, one of the main purposes of Table 23 is to facilitate the specification of contractual requirements for contractors. Therefore the focus of this classification is on the description of outcomes rather than on methods of implementation or application. In other words Table 23 specifies what products are to become part of the building, not how they are to be built, and much less how they are supposed to be used.

Yet, even this classification criterion not consistent either. Because Table 22 also attempts to serve as source for specification of technical requirements and cost information for construction, its scope had to be expanded to include the description of construction processes and activities required to achieve such work results.

Thus, entries in the Table 22 - Works Results refer sometimes to processes, procedures and activities required to achieve work results, and sometimes refer to work results themselves as final outcome of such processes. For example, under the Level 2 Title: "220-04 05 00 Common Work Results for Masonry", a series of masonry specific construction activities are referred by name. These include various types of mortaring, grouting, anchorage and reinforcing activities. The fact that each type requires a different set of skills, equipment and procedures is not made explicit, thus requiring interpretation and agreement.

In other cases however, interpretation and agreement may be harder to achieve. This is the case when work results are referred to by the name of a product rather than the process required. In these situations, not only construction process is implicit in the description, but also the work result and product application is not clear. For example, under Level 2 Title: "Clay Unit Masonry", certain generic unit types are listed, such as brick masonry, clay tile masonry or fluted concrete unit masonry among others without further explanation. While there are typical applications for each of those, it may not always be the case. For innovative applications, different construction processes may be required, involving different skill sets, trade coordination efforts, resources, etc.

Yet, in some other cases the products being referred are not even work results from construction process, but construction equipment such as trucks and other types of vehicles.

Table 36 shows a sample of OmniClass^{TM} Table 22 - Work Results illustrating some of these problems.

Table 36: Appendix A: Sample of Work Results. OmniClass Table 22

OmniClass™ Table 22 - Work Results		
Number	**Title**	**Definitions**
22-02 51 00	**Physical Decontamination**	None.
22-02 51 29 13	High-Pressure Water Cleaning Decontamination	None.
22-04 05 00	**Common Work Results for Masonry**	None.
22-04 05 13 26	Engineered Masonry Mortaring	None.
22-04 22 00	**Concrete Unit Masonry**	None.
22-04 22 23 16	Fluted Concrete Unit Masonry	None.
22-03 81 00	**Concrete Cuttings**	None.
22-03 81 23	Hand Concrete Wall Sawing	None.
22-07 42 00	**Wall Panels**	None.
22-07 42 13 19	Insulated Metal Wall Panels	None.
22-41 62 00	**Trucks**	None.
22-41 62 13	Cement Mixer Trucks	None.
22-41 62 00	**General Vehicles**	None.
22-41 62 13	Bicycles	None.

A.4 OmniClass^{TM} Table 21 - Elements

The definition of Element in the Table 21 of OmniClass^{TM} is as follows: "An Element is a major component, assembly, or "construction entity part which, in itself or in combination with other parts, fulfills a predominating function of the construction entity" (ISO 12006-2). Predominating functions include, but are not limited to, supporting, enclosing, servicing, and equipping a facility. Functional descriptions can also include a process or an activity... A Designed Element is an "Element for which the work result(s) have been defined." (ISO 12006-2)."

In this way, elements, particularly *designed* elements, constitute a general description intended as input for a more detailed specification of work results using Table 23. Such relationship between tables is made explicit in Table 21 by indexing Table 23 entry numbers.

However, since a building element may participate in the satisfaction of several different functions, the classification of elements based on a single predominating function seems problematic. Instead, semi-lattice representations are needed to capture the many-to-many mapping of functional relationships, especially the ones that are relevant at the whole systems level. Unfortunately these are not possible in the current hierarchical classification model adopted by OmniClass^{TM} tables. As result, elements are forced into individual categories that capture a single functional point of view. In situations when this becomes overly restricted, elements can also be described in other tables in order to cover for additional functional aspects (e.g. Table 23 - Products).

Regarding the IFC conceptualization, Table 21 - Elements also introduces some conflicting views and definitions. Here, entire systems are described as "elements", as opposed to groups in IFC, which only considers the components of systems as "elements". The fact that systems are important results of construction work, as discussed in section 3.3.2, is acknowledged in OmniClass^{TM} by explicitly relating elements to entries in Table 22 - Work Results.

In some situations, the notion of "functional systems" is implicit in the main category title (i.e. Level 1 Title) or the element title (i.e. Level 4 Title). Such is the case for the element "Exterior Wall Construction" under the general category of "Shell". Yet, the exact

Table 37: Appendix A: Sample of Elements. OmniClass Table 21

OmniClass™ Table 21 - Elements		
Number	**Title**	**Work Results Number**
21-01 00 00	**Substructure**	
21-01 10 20 30	Special Foundation Walls	22-31 66 16
21-02 20 10 20	Exterior Wall Construction	None
21-01 40 90 20	Vapor Retarder	22-07 26 00
21-02 00 00	**Shell**	
21-02 10 80 80	Ladders	22-05 51 23
21-02 20 10 10	Exterior Wall Veneer	None
21-02 20 10 20	Exterior Wall Construction	None
21-02 20 10 50	Parapets	None
21-04 00 00	**Services**	
21-04 20 10	Domestic Water Distribution	22-22 11 00
21-04 60 30 10	Audio-Video Systems	22-27 41 00
21-04 20 30 20	Stormwater Drainage Piping	22-22 14 13

predominant function of "Shell" is not made clear. In other situations, the predominant function of a system is implied by reference to Table 22- Work Results. This is the case of the element "21-04 40 10 50 Fire-Extinguishing", which refers to work result "22-21 20 00 Fire-Extinguishing Systems" without further specification.

MOrevoer, the analysis of element titles in Table 21 indicates that the use of predominant functions as criterion can only be applied in a convincing manner at the lowest levels of the classification, when systems and components described are predominantly mono-functional. Unfortunately, these represent only a subset of all systems and components that comprise the built environment.

For higher levels of the classification the criterion is more ambiguous and clearly multi-functional. For instance the category "Substructure" comprises functional elements as diverse as foundations, subgrade enclosures, insulation, vapor retarders and drainage, to name a few. Table 37 summarizes some of these inconsistencies.

A.5 OmniClass^{TM} Table 13 - Spaces by Function

Spaces by Function in are defined as "basic units of the built environment delineated by physical or abstract boundaries, and characterized by their function or primary use." Some examples provided are kitchens, elevator shafts, office spaces and and even sidewalks. The classification of spaces using form as criterion is developed in Table 14 - Spaces by Form. Both tables are intended to provide two independent but complementary ways of characterizing spaces.

In this table the problem of properly characterizing functionality, particularly for entities that are fundamentally multi-functional emerges again. In order to fit such entities into a single hierarchical classifications, a series of implicit assumptions and inconsistent definitions are made, similarly to other tables discussed previously.

Table 38 shows few examples. In some cases, the classification often seems to be based on parthood or composition, rather than function. This situation happens in cases of spaces working as a functional parts of a larger space unit type. For example, "Exterior Parking Circulation" are defined under "Exterior Parking Spaces" because they are a necessary part. In other cases classification is based on the materials used in one of the bounding or supporting elements of the space. For instance, under "Recreation Spaces", there are entries for "Baseball Field", "Grass Playing Fields" and "Synthetic Fields".

In other cases, circulation of people, goods, or utilities seems to be the predominant criterion for characterization of spaces. Thus, a number of mechanical circulation spaces are defined as types of "Facility Service Spaces", under the Level 2 Title - "Vertical Penetration", even though many mechanical circulation spaces are not necessarily "vertical penetrations", do not require "vertical penetrations" nor are part of " vertical penetrations". This would certainly be the case for "Elevator Shaft", but not for "Elevator Machine Room". Other entities defined as mechanical circulation spaces are not even spaces at all, but transportation devices. For instance, "Dumbwaiter", "Escalator" and "Freight Elevator".

It could be argued that the spaces required for each of those transportation devices are implied in the titles and textual definitions provided. However this leads to ambiguity

with other definitions provided in other OmniClass^{TM} tables as well as in IFC. For instance "Power Distribution Network" is defined in Table 13 as a service space, whereas "Power Distribution" is a "Services Element" in Table 21. In IFC they would be both considered as groups, sub-typed by *IfcSystem* [48].

Additionally, under the same general category of "Facility Service Spaces", there are definitions for other types of circulation spaces such as "Ramps" and "Stairways". However, other types circulation spaces are defined under the category "Circulation Spaces", which includes "Corridors", "Aisle" and Vestibule". This last category also includes "Landing", which is both a structural and functional part of staircases and stairways in general, which in turn are defined elsewhere. Counter-intuitively, "Refuge Spaces" are also defined as a type of "Circulation Spaces".

Table 38: Appendix A: Sample of Spaces by Function. OmniClass Table 13

OmniClass™ Table 13 - Spaces by Function		
Number	**Title**	**Definition**
13-21 00 00	**Parking Spaces**	Spaces used to circulate and station vehicles.
13-21 11 11	Exterior Parking Circulation	Outdoor space used to circulate vehicles and providing access to parking stalls.
13-23 00 00	**Facility Service Spaces**	Portion of a building that provides services that enable occupants to work in a building.
13-23 11 11	Mechanical Circulation	Space used by mechanical modes of circulation such as elevators and escalators providing transportation between floors of a structure.
13-23 11 13	Stairway	Space used by a static circulation path providing transportation between floors of a structure.
13-23 11 15	Monumental Stair	Space occupied by a larger than necessary, architectural stair. Space with clear headroom under the stair may be classified differently.
13-23 11 19	Chimney	A primarily vertical enclosure containing one or more passageways for conveying flue gasses to the outside atmosphere.
13-25 00 00	**Circulation Spaces**	Spaces for circulation that provide or control access to and between other spaces within the facility, entry, and egress.
13-25 11 11	Corridor	An enclosed exit access component that defines and provides a path of egress travel to an exit.
13-25 13 13	Entry Lobby	A large entrance area of hall that serves as a *foyer*.
13-25 13 23	Landing	An in-between platform or large bottom-most or top-most step of a staircase.
13-25 23 00	Refuge Space	An enclosed space that is protected from the effects of fire permitting a delay in required egress travel time.
13-33 00 00	**Recreation Spaces**	space for any leisure activity, such as play, that diverts, amuses or stimulates.
13-33 11 13 17	Grass Playing Fields	A grass field on which a game, esp. a ball game, is played.
13-33 11 13 19	Synthetic Fields	A field often used for team sports, planted with a synthetic grass substitute.

A.6 OmniClass™ Table 49 - Properties

Properties are defined in OmniClass™ Table 49 as characteristics of construction entities. They obtain meaning through reference to one or more construction objects to which they may be applied.

In this way properties are considered dependent entities, which characterize OmniClass™ entries in a variety of ways. Some examples provided include color, width, length, thickness, diameter, areas, fire resistance rating, weight, compressive strength, etc. Given the infinite list of possible properties, Table 49 focuses on providing a hierarchical classification for only the most common property types used in construction.

A first observation from the definition above is the statement about properties as characteristics (i.e. characterizing) of construction entities. Interestingly, functions are also viewed as the defining *characteristics* of both elements and spaces in their respective OmniClass™ tables. Since the relevance of functions in the context of Building Design, construction, operations and maintenance, it makes sense to address them separately. The point to be made however, is that functions are seem in OmniClass™, albeit implicitly, as properties of construction entities. Yet, the precise ontological nature of different functional properties is not addressed in OmniClass™, nor in other classification systems and information models such as STEP and IFC.

In particular, the treatment of functions as behavioral abstractions is not recognized or addressed, even informally. In this way the classification strategy for objects, processes and properties in all these standards results haphazard and inconsistent, hindering the pragmatic need of data integration, interoperability and automation.

This is especially evident in the case of Table 49, where there is no clear distinction between structural, functional and behavioral properties.

The category "Identification Properties" is intended to provide meta-data about objects mostly for administrative purposes. These include names and numbers to identify physical addresses of facilities, rooms in a building, objects and equipment types, owner, fabricator and emergency contact information, among others. Nevertheless, some other identification properties are set under separate categories such as "Location Properties" and "Source

Properties". The first comprises concepts as diverse as "Political / Legal Locations" and "Manufacturing and Product Locations".

The main problem is that all of these supposed locations are not locations at all, but other type of entities that may exist or act on certain geographic locations. It is the meaning of each of those entities, in terms of their administrative *purpose*, that makes them different from each other, not their location. In other words, while a manufacturing district may exist in the same geographic location of a municipal district, they are clearly not the same thing [66].

Regarding "Source Properties", this category comprises identification information pertaining to manufactured or construction products (i.e. work results). This includes names for manufacturer, designer and product, product packaging, product features and accessories. It also includes property titles that one would expect in other categories, such as "Color", "Finish", "Installation Method" or "Ambient Temperature during Installation".

Tables 39 and 40 provide some examples of these properties and the problems of inconsistency discussed so far. Notice the inclusion of a reference to IFC along with occupant demographics as identification properties. Additionally, latitude and longitude, country, Authority Having Jurisdiction (AHJ), ceiling, wall and floor finishes are all defined as location properties.

Table 39: Appendix A: Sample of Identification Properties. OmniClass Table 49

OmniClass™ Table 49 - Properties		
Number	**Title**	**Definition**
49-11 00 00	**Identification Properties**	Properties that identify objects or provide or enhance meta-data about objects.
49-11 11 00	**Facility Identifications**	Properties that identify a facility by location or other criteria. Refer to Tables 11 and 12 for Facility Types.
49-11 11 19	Facility Name	A word or set of words a building is known by.
49-11 11 23	Street Address	Number of a specific building on a specified street name, typically assigned by the post office and visible at main entries.
49-11 31 00	**Space Identifications**	Properties that identify or describe volumes enclosed by surfaces. Refer to Tables 13 and 14 for Space Types.
49-11 31 15	Standard Space Name	Generic room or area name.
49-11 31 19	Space Number	Room code visible to public, typically on door signs or floor plans.
49-11 00 00	**Occupancy Identifications**	Properties that identify or describe facility occupants and user groups.
49-11 51 11	Occupancy Class	General categories of structures based on use and hazard, typically defined in local building codes.
49-11 51 15	Occupant Demographics	Statistical data relating to the population and particular groups that typically use a building or site.
49-11 71 00	**Work Result Identifications**	Properties that identify or describe work results or details associated with work results.
49-11 71 21	Industry Foundation Class (IFC)	A class within the IFC data model to describe building and construction industry data.
49-11 71 43	Approved By	Person or organization who approves products or information related to a particular project.
49-11 71 45	Designed By	Person or organization who designs products or information related to a particular project.

Table 40: Appendix A: Sample of Location Properties. OmniClass Table 49

OmniClass™ Table 49 - Properties		
Number	**Title**	**Definition**
49-21 00 00	**Location Properties**	Properties that provide physical location information.
49-21 11 00	**Geographic Locations**	Properties that describe positions or points in physical space that an object occupies on the Earth's surface.
49-21 11 11	Latitude	Angular distance north or south of Earth's equator.
49-21 11 13	Longitude	Angular distance on Earth's surface, measured east or west from the prime meridian near Greenwich, England, to the meridian passing through a position.
49-21 51 00	**Political / Legal Locations**	Properties that describe positions in relation to nations, national subdivisions, and their relationship to the world.
49-21 51 11	Country	Value representing a country of origin or location.
49-21 51 19	Municipality	Unique Community ID Number or Geocode. Includes cities, towns, neighborhoods and other local areas.
49-21 51 23	Authority Having Jurisdiction (AHJ)	The governmental agency or sub-agency which regulates the construction process. In most cases, this is the municipality in which the building is located.
49-21 51 00	**Manufacturing and Product Locations**	Properties that describe locations in relation to production and distribution of objects or resources.
49-21 71 11	Manufacturer Location	Geographic coordinates, typically relative to a project site, where products are manufactured for a project. May contribute to sustainable design documentation.
49-21 71 15	Warehouse Location	Geographic coordinates where products are stored by manufacturers, contractors, or owners until sale or installation.
49-21 91 00	**In-Building Locations**	Properties that describe locations in relation to production and distribution of objects or resources.
49-21 91 11	Ceiling Finish	None.
49-21 91 13	Wall Finish	None.
49-21 91 15	Floor Finish	None.

More significant issues with classification however occur under the categories "Physical Properties", "Performance Properties" and "Properties of Facility Services". A main reason is the *use-mention distinction* problem. This happens in classification systems when different contexts of meaning are treated as they were the same [286]. In this particular case, the problem occurs when observable aspects of reality that are supposed to be the subject matter of the classification (e.g. physical properties to be measured and used) are confused with the abstractions adopted to describe them (e.g. systems of representation and measurement).

Examples of this in the category "Physical Properties" occur under the "Quantity Properties" sub-category. This comprises entities such as "Unit of Measurement", "Capacity", "Mean Size" or "Clearance Requirements" which do not refer to physical properties themselves, but to some mathematical abstractions chosen describe them, namely, "count or number" [sic].

Similarly, "Shape Properties", also classified under "Physical Properties", are rather mathematical abstractions, for which no particular physical object needs to be alluded. Because of this independence from concrete referents, other abstract aspects such as "Single Dimensions" and "Area Dimensions"can be represented. Such representations however depends primarily on the purpose at hand. Unfortunately, because such purposes or intended applications are never explicitly addressed as part of the classification criteria, sub-categories of "Physical Properties" such as "Area Dimensions" end up comprising concepts as dissimilar as"Net Rentable Area", "Floor Area Per Occupant", "Inside Diameter", "Perimeter", "Plane Angle" and "Rise".

There are many other similar semantic inconsistencies across categories and sub-categories in Table 49. Because of the lack of proper use-mention distinction, concepts such as rotational frequency, speed and velocity are defined as "Relational Measurements", under the "Physical Property" category. Yet, other rates and relational measurements are described under the category "Performance Properties".

In this last case properties often refer to behavioral aspects of systems, products and materials. Exceptions do occur though, especially when structural properties are added

to the mix. Examples are “Dimensional Tolerance”, “Shape Tolerance“, “Plumbness” and “Squareness” and the like. Each of these contains its own set of internal inconsistencies, which are not going to be addressed here.

Additional inconsistency problems caused by the use-mention distinction in the “Performance Properties” category are the inclusion of test methods, test authorities, inspection protocols and reference standards of various kinds. Generally speaking, all of those concepts are concerned with the administrative processes of specification, verification and validation of performance, rather than being performance properties themselves.

Perhaps the most problematic sub-category under “Performance Properties“is the “Function and Use Properties”. This category is intended to define various functional aspects of objects during service conditions. It comprises concepts such as “Functional Efficiency”, “Functional Limitations”, “Methods of Operation”, “Functional Capacity”, “Code Performance”, “Ease of”, “Serviceability”, “Suitability”, “Workability”, “Weatherability” and “Waste Produced in Use”. It could be argued that some of these are very closely related to sustainability and Life Cycle Analysis properties.

Curiously enough, the latter are defined under “Physical Properties” and not under “Performance Properties”. The same applies to “Temperature Ranges”, “Indoor Air Quality (IAQ)”, “Torque” and “Properties of Structural Loading” among others. The sub-category “Strength Properties”, which comprises “Adhesion Strength”, “Compressive Strength” and “Flexural Strength” are all defined in turn under “Performance Property” and not under “Physical Property”.

Finally, many other performance properties are defined under the category “Properties of Facility Services”. These range from properties of building systems such as HVAC efficiency and loads, electrical properties such as amperage, electric conductivity, grounding resistance and impedance, to various properties related to lighting systems, like glare, light transmission, exposure, absorption and reflectance among others.

The following tables 41, 42, 43, and 44 illustrate some of these problems in each of the categories and sub-categories discussed.

Notice that, besides the problem of use-mention distinction, there are several redundant

definitions. For example, entries 49-81 31 11 “Adhesion Strength” and 49-81 31 21 “Bond Strength” under the “Strenght Properties” category refer to the same concept, whereas “Animal Resistance” could comprise the concept of resistance against bacteria, fungus, mildew, and thermites under a hierarchy, instead of having them all at the same level. On the other hand, entries 49-81 21 31 “Weatherability”, 49-81 31 53 “Impact Strength” and 49-81 31 55 “Resistance to Intentional Attack” seem rather misplaced by not being considered as “Durability Properties”.

Table 44 shows how the category “Properties for Facility Services” refers to properties of building systems and other components from the perspective of operations and maintenance. Arguably, many of these properties are indeed related to performance or methods of performance evaluation that should have been described under different categories, for instance, under “Performance Properties”. Other properties in turn refer to product and element types better addressed by other tables altogether. Therefore most of the category “Properties for Facility Services” seems redundant and unnecessary.

Table 41: Appendix A: Sample of Physical Properties. OmniClass Table 49

Number	Title	Definition
49-71 00 00	**Physical Properties**	Properties that describe the physical presentation of an object or that exist within an object by itself or as used, considered, or installed.
49-71 11 00	**Quantity Properties**	Properties that describe the count or number of components.
49-11 11 19	Unit of Measurement	Refers to the way in which the amount of a substance or item can be broken down and counted in a uniform manner.
49-11 11 23	Capacity	Refer to Function and Use and Facility Services for equipment capacities.
49-71 15 00	**Shape Properties**	Properties that describe the geometric shape of component.
49-71 15 13	Shape	Describes the geometric layout or configuration of an object.
49-71 15 15	Geometry	Describes the shape and form of an object, usually with respect to mathematical shapes.
49-71 23 00	**Area Dimensions**	Properties that quantify the extents of spaces with a 2 dimensional bound.
49-71 23 15	Net Rentable Area	Measurement of the area in a building, usually expressed in square footage, that can be leased or rented to occupants.
49-71 23 25	Circumference	Measurement of the outside edge of a circle.
49-71 31 00	**Relational Measurements**	Properties that quantify one measurement in relation to another.
49-71 31 11	Acceleration	Measurement of the increase in velocity of a moving object.
49-71 31 39	Unit of Analysis	The major entity that is being analyzed in a study.
49-71 35 00	**Properties of Sustainability**	Attributes describing concept of balance between resource consumption and regeneration.
49-71 35 13	Harvest Method	Procedures to gather materials, usually biological, for use in construction materials, especially crops and timber.
49-71 35 27	Life Cycle Analysis (LCA)	Standard self-certification to be verified by an independent third party.
49-71 57 00	**Properties of Air and Other Gases**	Properties that pertain to gasses in the atmosphere.
49-71 57 11	Indoor Air Quality (IAQ)	A measure of the purity of air in a given space.
49-71 57 25	Visibility	The measure of the distance at which an object or light can be clearly discerned.

Table 42: Appendix A: Sample of Performance Properties. OmniClass Table 49-A

Number	Title	Definition
49-81 00 00	**Performance Properties**	Properties that express the behavior of an object in reaction to physical properties and forces.
49-81 11 00	**Testing Properties**	Properties that define testing methods and conditions.
49-81 11 11	Test Method	Describes the type of test to be performed.
49-81 15 00	**Tolerance Properties**	Properties that define permissible limits or variations in limit of dimensional and other measured properties.
49-81 15 11 7	Deflection Tolerance	Measurement of an object. . . to withstand deflection erosion or break down.
49-81 15 15	Shape Tolerance	A range of acceptable deviations from the creation intent of an item.
49-81 15 21	Squareness	Shaped like a square.
49-81 21 00	**Function and Use Properties**	Properties that define aspects of functionality and characteristics of the object for a particular service.
49-81 21 13	Functional Limitations	Circumstances, restrictions, or constraints on the ability to perform a regular function.
49-81 21 15	Method of Operation	Manner in which a product or system is to be used.
49-81 21 23	Ease of	Measurement of the ability to use something, often the inverse of the difficulty.
49-81 21 31	Weatherability	Resistance properties for heat, moisture, cold, solar radiation, etc.
49-81 31 00	**Strength Properties**	Properties that define ability of an object to resist applied force.
49-81 31 11	Adhesion Strength	Measurement of the amount of force a bonding agent can withstand.
49-81 31 27	Compressive Strength	Ability to withstand crushing force.
49-81 31 21	Bond Strength	Measurement of the amount of force an adhesive can take before giving way.
49-81 31 49	Friction	Measurement of the resistance created when one object moves or slides against another.
49-81 31 55	Resistance to Intentional Attack	Measurement of the ability to withstand damage or theft.

Table 43: Appendix A: Sample of Performance Properties. OmniClass Table 49-B

Number	Title	Definition
49-81 00 00	**Performance Properties**	Properties that express the behavior of an object in reaction to physical properties and forces.
49-81 41 00	**Durability Properties**	Properties that define ability of an object to maintain resistance to the effects of applied force.
49-81 41 11	Abrasion Resistance	Able to withstand force or friction.
49-81 41 19	Animal Resistance	Able to withstand attacks, penetration, and habitation by various types of fauna.
49-81 41 21	Bacteria Resistance	Able to withstand microorganisms.
49-81 41 25	Corrosion Resistance	Able to withstand salt water, dissimilar metals, rust, oxidation and similar effects.
49-81 41 49	Insect Resistance	Will not support growth or life of bugs.
49-81 41 51	Lubricity	Slippery or smooth.
49-81 41 67	Termite Resistance	Treatment, typically for wood, to prevent infestation.
49-81 51 00	**Combustion Properties**	Properties that define the burning or oxidation of a substance, in gaseous, liquid, or solid form and the results of such burn.
49-81 51 13	Combustibility	Capable of igniting and burning.
49-81 51 15	Fire Resistance Rating	The duration a material or system can withstand standard tests.
49-81 51 43	Fire Resistance	Refers to the degree of which something is able to resist burning or ignition.
49-81 61 00	**Properties of the Envelope**	Properties Between Inside and Outside Environment. Refer to other categories for Structural, Moisture/Permeability, Combustion and Fire Resistance, Acoustics, Impact Resistance.
49-81 61 13	Air Infiltration	Measurement of the ability of air to permeate a given area.
49-81 61 15	Air Tightness	Not allowing air to pass through or permeate.
49-81 71 00	**Properties of Permeability and Moisture Resistance**	Properties of objects, surfaces or membranes that define the characteristics and the degree to which it can be pervaded by a liquid or gas (as by osmosis or diffusion).
49-81 71 17	Gas Permeability	Ability of gasses to seep into the system.
49-81 71 29	Porosity	The ratio of the total amount of void space in a material (due to poses, small channels, and so on) to the bulk volume occupied by the material.

Table 44: Appendix A: Sample of Properties for Facility Services. OmniClass Table 49

Number	Title	Definition
49-91 00 00	**Properties of Facility Services**	Properties of systems used in facility services, often comprised of a number of individual component objects – these properties tend to apply to whole systems, component properties are typically classified elsewhere.
49-91 11 00	**General Properties of Facility Services**	General properties defining aspects of systems for Facilities Services.
49-91 11 13	Area Covered	A two dimensional metric or a percentage of a total area.
49-91 11 23	Parts	Items which can be a larger, more useful object.
49-11 71 45	Designed By	Person or organization who designs products or information related to a particular project.
49-91 11 53	Number of Elements	A numerical value of the number of distinct elements in a system.
49-91 25 00	**Properties of HVAC Systems**	System or component properties uniquely found in HVAC Systems.
49-91 25 13	HVAC Efficiency	Describes the energy efficiency of the HVAC system.
49-91 25 15	HVAC Loads	Describes the total draw on an HVAC system.
49-91 41 00	**Properties of Electrical Systems**	Properties defining supply, demand and characteristics of Electrical Power, features of Electrical Systems and Electrical Components.
49-91 41 11	Amperage	Refers to the flow of electric charge through a medium, measured in amperes.
49-91 41 33	Earth Ground Resistance	Denotes the resistance of earth to electricity.
49-91 41 45	Electrical Conductivity	Denotes how easily something conducts electricity.
49-91 51 00	**Properties of Lighting Systems**	Properties defining Lighting Systems or Lighting Components.
49-91 51 11	IEEE Illumination Levels	Groups of lighting organized by intensity.
49-91 51 17	Glare	Minimum amount of glare to trigger the shades to close.
49-91 51 19	Glare Index	A rating that denotes the presence of inhibiting or disruptive glare.
49-91 51 29	Lighting Calculation Workplane	The level used as the base for calculating Illumination.
49-91 51 47	Lighting Controls	Describes the types of controls to manage illumination.

APPENDIX B

ASPECTO-FR ONTOLOGY

Prefix: : <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#>

Prefix: DOLCE−Lite: <http://www.loa−cnr.it/ontologies/DOLCE−Lite.owl#>

Prefix: aspecto−fr: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#>

Prefix: dc: <http://purl.org/dc/elements/1.1/>

Prefix: dolce−lite−fr: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#>

Prefix: ns: <http://creativecommons.org/ns#>

Prefix: owl: <http://www.w3.org/2002/07/owl#>

Prefix: rdf: <http://www.w3.org/1999/02/22−rdf−syntax−ns#>

Prefix: rdfs: <http://www.w3.org/2000/01/rdf−schema#>

Prefix: swrl: <http://www.w3.org/2003/11/swrl#>

Prefix: swrla: <http://swrl.stanford.edu/ontologies/3.3/swrla.owl#>

Prefix: swrlb: <http://www.w3.org/2003/11/swrlb#>

Prefix: vann: <http://purl.org/vocab/vann/>

Prefix: xml: <http://www.w3.org/XML/1998/namespace>

Prefix: xsd: <http://www.w3.org/2001/XMLSchema#>

Ontology: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR>

Import: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl>

AnnotationProperty: rdfs:comment

AnnotationProperty: swrla:isRuleEnabled

Datatype: rdf:PlainLiteral

Datatype: xsd:boolean

Datatype: xsd:string

ObjectProperty: <http://www.ou.edu/coa/ontologies/2018/4/Aspecto-FR-Coworking#functional_role_in>

SubPropertyChain:

aspecto-fr:_c_causes o dolce-lite-fr:proper-part-of o aspecto-fr:r_outpd

ObjectProperty: aspecto-fr:_AS_aspect_of_function

ObjectProperty: aspecto-fr:_AS_co-participant

ObjectProperty: aspecto-fr:_AS_environment

ObjectProperty: aspecto-fr:_DF_artifact

InverseOf:

aspecto-fr:nominal_device_function

ObjectProperty: aspecto-fr:_DF_by_behavior

Domain:

aspecto-fr:_D_function

Range:

aspecto-fr:EPD

ObjectProperty: aspecto-fr:_DF_inPD

SubPropertyChain:

aspecto-fr:b0 o aspecto-fr:behavior_constraint_on_perdurant o aspecto-fr:r_epd

Range:

aspecto-fr:InPD

ObjectProperty: aspecto-fr:_DF_outPD

SubPropertyChain:

aspecto−fr:b1 o aspecto−fr:behavior_constraint_on_perdurant o aspecto−fr:r_epd

Range:

aspecto−fr:OutPD

ObjectProperty: aspecto−fr:_DF_satCrBeh

ObjectProperty: aspecto−fr:_DF_stakeholder

ObjectProperty: aspecto−fr:_EF_aspect_system

ObjectProperty: aspecto−fr:_EF_inPD

ObjectProperty: aspecto−fr:_EF_in_environment

ObjectProperty: aspecto−fr:_EF_in_mode_of_deployment

ObjectProperty: aspecto−fr:_EF_nominal_artifact

SubPropertyChain:

aspecto−fr:b0 o aspecto−fr:_DF_artifact

ObjectProperty: aspecto−fr:_EF_outPD

SubPropertyChain:

aspecto−fr:b1 o aspecto−fr:r_function o aspecto−fr:r_epd

ObjectProperty: aspecto−fr:_EF_stakeholder

ObjectProperty: aspecto−fr:_MD_causal_model

ObjectProperty: aspecto−fr:_MD_causal_model_InPD

ObjectProperty: aspecto−fr:_MD_causal_model_OutPD

ObjectProperty: aspecto−fr:_b_constrained_participant_in

SubPropertyOf:

aspecto−fr: _b_qualified_participant_in ,

dolce−lite−fr:participant−in

ObjectProperty: aspecto−fr:_b_has_capability

SubPropertyChain:

dolce−lite−fr:proper−part o aspecto−fr:_b_has_capability

SubPropertyChain:

inverse (aspecto−fr:_DF_artifact) o aspecto−fr:b1

InverseOf:

aspecto−fr:behavior_constraint_on_endurant

ObjectProperty: aspecto−fr:_b_qualified_participant_in

SubPropertyOf:

dolce−lite−fr:participant−in

SubPropertyChain:

aspecto−fr:_b_has_capability o aspecto−fr:behavior_constraint_on_perdurant o aspecto−fr:
r_epd

ObjectProperty: aspecto−fr:_c_caused_by

Characteristics:

Transitive

Domain:

aspecto−fr:EPD

Range:

aspecto−fr:EPD

InverseOf:

aspecto−fr:_c_causes

ObjectProperty: aspecto−fr:_c_causes

SubPropertyChain:

aspecto−fr: r_life o dolce−lite−fr:proper−part o aspecto−fr:_d_causes

Characteristics:

Transitive

Domain:

aspecto−fr:EPD

Range:

aspecto−fr:EPD

InverseOf:

aspecto−fr:_c_caused_by

ObjectProperty: aspecto−fr:_d_caused_by

SubPropertyOf:

aspecto−fr:_c_caused_by

InverseOf:

aspecto−fr:_d_causes

ObjectProperty: aspecto−fr:_d_causes

SubPropertyOf:

aspecto−fr:_c_causes

Characteristics:

Functional

InverseOf:

aspecto−fr:_d_caused_by

ObjectProperty: aspecto−fr:_n_caused_by

SubPropertyOf:

aspecto−fr:_c_caused_by

Domain:

aspecto−fr:InPD

Range:

aspecto−fr:OutPD

InverseOf:

aspecto−fr:_n_causes

ObjectProperty: aspecto−fr:_n_causes

SubPropertyOf:

aspecto−fr:_c_causes

InverseOf:

aspecto−fr:_n_caused_by

ObjectProperty: aspecto−fr:b0

Domain:

aspecto−fr:_P_causal_pattern

ObjectProperty: aspecto−fr:b1

Domain:

aspecto−fr:_P_causal_pattern

ObjectProperty: aspecto−fr:behavior_constraint_on_endurant

InverseOf:

aspecto−fr:_b_has_capability

ObjectProperty: aspecto−fr:behavior_constraint_on_perdurant

Domain:

aspecto−fr:Behavior

ObjectProperty: aspecto−fr:behavioral_constraint_on

ObjectProperty: aspecto−fr:causal_participant

InverseOf:

aspecto−fr:causal_participant_in

ObjectProperty: aspecto−fr:causal_participant_in

SubPropertyChain:

dolce−lite−fr:participant−in o aspecto−fr:_c_causes

InverseOf:

aspecto−fr:causal_participant

ObjectProperty: aspecto−fr:has_component

Domain:

aspecto−fr:material−artifact

ObjectProperty: aspecto−fr:member_of_aspect_system

ObjectProperty: aspecto−fr:nominal_device_function

Range:

aspecto−fr:_D_function

InverseOf:

aspecto−fr:_DF_artifact

ObjectProperty: aspecto−fr:nominal_participant

SubPropertyOf:

dolce−lite−fr:participant

InverseOf:

aspecto−fr:nominal_participant_in

ObjectProperty: aspecto−fr:nominal_participant_in

SubPropertyOf:

dolce−lite−fr:participant−in

SubPropertyChain:

inverse (aspecto−fr:_DF_artifact) o aspecto−fr:_DF_outPD

Domain:

inverse (aspecto−fr:_EF_nominal_artifact) some aspecto−fr:_E_function

Range:

aspecto−fr:EPD

InverseOf:

aspecto−fr:nominal_participant

ObjectProperty: aspecto−fr:r_artifact

ObjectProperty: aspecto−fr:r_behavior

ObjectProperty: aspecto−fr:r_environment

SubPropertyOf:

owl:topObjectProperty

ObjectProperty: aspecto−fr:r_epd

ObjectProperty: aspecto−fr:r_function

ObjectProperty: aspecto−fr:r_inpd

ObjectProperty: aspecto−fr:r_life

ObjectProperty: aspecto−fr:r_outpd

ObjectProperty: aspecto−fr:s_adjacent_to

Domain:

dolce−lite−fr:physical−endurant

Range:

dolce−lite−fr:physical−endurant

ObjectProperty: aspecto−fr:s_connected_to

SubPropertyOf:

dolce−lite−fr:weak−connection

Domain:

dolce−lite−fr:physical−object

Range:

dolce−lite−fr:physical−object

ObjectProperty: aspecto−fr:s_directly_connected_to

SubPropertyOf:

aspecto−fr:s_connected_to

ObjectProperty: dolce−lite−fr:life

SubPropertyChain:

dolce−lite−fr:part o dolce−lite−fr: life

SubPropertyChain:

dolce−lite−fr:proper−part o dolce−lite−fr: life

ObjectProperty: dolce−lite−fr:part

Characteristics:

Transitive ,

Reflexive

ObjectProperty: dolce−lite−fr:part−of

Characteristics:

Transitive ,

Reflexive

ObjectProperty: dolce−lite−fr:participant

ObjectProperty: dolce−lite−fr:participant−in

SubPropertyChain:

dolce−lite−fr: life o dolce−lite−fr:part o aspecto−fr:r_outpd

SubPropertyChain:

inverse (aspecto−fr:_EF_nominal_artifact) o aspecto−fr:_EF_outPD

ObjectProperty: dolce−lite−fr:proper−part

ObjectProperty: dolce−lite−fr:proper−part−of

ObjectProperty: dolce−lite−fr:weak−connection

ObjectProperty: owl:topObjectProperty

DataProperty: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR−KB#EF_realizability
>

Class: aspecto−fr:APD

SubClassOf:

aspecto−fr:EPD,

aspecto−fr:_c_causes only aspecto−fr:APD

Class: aspecto−fr:Actual_life

SubClassOf:

aspecto−fr:EPD,

aspecto−fr:r_life some Self

Class: aspecto−fr:Air

SubClassOf:

aspecto−fr:non−agentive−physical−object

Class: aspecto−fr:Aspect_system

SubClassOf:

aspecto−fr:Environment

Class: aspecto−fr:Behavior

SubClassOf:

dolce−lite−fr:quality ,

aspecto−fr:behavior_constraint_on_perdurant some aspecto−fr:EPD

Class: aspecto−fr:CauseMD

SubClassOf:

aspecto−fr:EPD

Class: aspecto−fr:EPD

EquivalentTo:

aspecto−fr:r_epd some Self

SubClassOf:

aspecto−fr:GEPD,

aspecto−fr:_c_causes only aspecto−fr:EPD,

dolce−lite−fr:proper−part only aspecto−fr:EPD

Class: aspecto−fr:Engineering_perdurant

SubClassOf:

dolce−lite−fr:perdurant

Class: aspecto−fr:Env1

SubClassOf:

aspecto−fr:Environment,

dolce−lite−fr:weak−connection only aspecto−fr:Environment

Class: aspecto−fr:Environment

EquivalentTo:

aspecto−fr:r_environment some Self

SubClassOf:

dolce−lite−fr:physical−object

Class: aspecto−fr:GEPD

SubClassOf:

aspecto−fr:Engineering_perdurant,

aspecto−fr:_c_causes only aspecto−fr:GEPD

Class: aspecto−fr:InPD

SubClassOf:

aspecto−fr:EPD,

aspecto−fr:r_inpd some Self

Class: aspecto−fr:Light−as−thing

SubClassOf:

dolce−lite−fr:amount−of−matter

Class: aspecto−fr:Light−transmittance−quality

SubClassOf:

dolce−lite−fr:quality−space

Class: aspecto−fr:Mode_of_deployment

SubClassOf:

aspecto−fr:EPD

Class: aspecto−fr:Objectified−relation

Class: aspecto−fr:OutPD

SubClassOf:

aspecto−fr:EPD,

aspecto−fr:r_outpd some Self

Class: aspecto−fr:Realizable_E_Function

EquivalentTo:

aspecto−fr:Satisfiable_E_function

and (<http://www.ou.edu/coa/ontologies/2018/6/Aspecto-FR-KB#EF_realizability>
value true)

Class: aspecto-fr:Reverberation-quality

SubClassOf:

dolce-lite-fr:quality-space

Class: aspecto-fr:Satisfiable_D_function

EquivalentTo:

aspecto-fr:_D_function
and (aspecto-fr:_DF_outPD some (aspecto-fr:_c_caused_by some (inverse (aspecto-fr:
_DF_inPD) some (aspecto-fr:_DF_artifact some (dolce-lite-fr:part some
((aspecto-fr:_b_has_capability some aspecto-fr:Behavior)
and (dolce-lite-fr:part-of some aspecto-fr:Technical_artifact)))))))

Class: aspecto-fr:Satisfiable_E_function

EquivalentTo:

aspecto-fr:_E_function
and (aspecto-fr:_DF_outPD some (aspecto-fr:_c_caused_by some (inverse (aspecto-fr:
_DF_inPD) some (aspecto-fr:_DF_artifact some (dolce-lite-fr:part some
((aspecto-fr:_b_has_capability some aspecto-fr:Behavior)
and (aspecto-fr:s_connected_to some (dolce-lite-fr:part-of value <http://www.ou.
edu/coa/ontologies/2018/6/Aspecto-FR-KB#Env1>)))))))

SubClassOf:

aspecto-fr:_E_function

Class: aspecto-fr:Smell-as-thing

SubClassOf:

dolce-lite-fr:amount-of-matter

Class: aspecto−fr:Sound−as−thing

SubClassOf:

dolce−lite−fr:amount−of−matter

Class: aspecto−fr:Sound−insulation−quality

SubClassOf:

dolce−lite−fr:quality−space

Class: aspecto−fr:Technical_artifact

SubClassOf:

aspecto−fr:material−artifact,

aspecto−fr:causal_participant_in some aspecto−fr:EPD

Class: aspecto−fr:Technical_system

SubClassOf:

aspecto−fr:material−artifact,

aspecto−fr:causal_participant_in some aspecto−fr:EPD,

inverse (aspecto−fr:_EF_nominal_artifact) some aspecto−fr:_E_function

Class: aspecto−fr:_Actual_behavior

SubClassOf:

aspecto−fr:_Possible_behavior

Class: aspecto−fr:_Behavior

SubClassOf:

aspecto−fr:Objectified−relation,

aspecto−fr:r_function some Self

Class: aspecto−fr:_D_function

SubClassOf:

aspecto−fr:_P_causal_pattern,

aspecto−fr:_DF_artifact only aspecto−fr: Technical_artifact ,

aspecto−fr:_DF_inPD only aspecto−fr:EPD,

aspecto−fr:_DF_outPD only aspecto−fr:EPD,

aspecto−fr:_DF_stakeholder only aspecto−fr:rational−physical−object

Class: aspecto−fr:_E_function

SubClassOf:

aspecto−fr:_P_causal_pattern,

aspecto−fr:_EF_inPD only aspecto−fr:EPD,

aspecto−fr:_EF_in_environment only aspecto−fr:Environment,

aspecto−fr:_EF_in_mode_of_deployment only aspecto−fr:Mode_of_deployment,

aspecto−fr:_EF_nominal_artifact only aspecto−fr: Technical_artifact ,

aspecto−fr:_EF_outPD only aspecto−fr:OutPD,

inverse (aspecto−fr:_AS_aspect_of_function) only aspecto−fr:Aspect_system

Class: aspecto−fr:_General_behavior

SubClassOf:

aspecto−fr:_Behavior

Class: aspecto−fr:_P_causal_pattern

SubClassOf:

aspecto−fr:Objectified−relation,

aspecto−fr:b0 some aspecto−fr:EPD,

aspecto−fr:b1 some aspecto−fr:EPD,

aspecto−fr:r_function some Self

Class: aspecto−fr:_Possible_behavior

SubClassOf:

aspecto−fr:_General_behavior

Class: aspecto−fr:agentive−physical−object

SubClassOf:

dolce−lite−fr:physical−object

Class: aspecto−fr:material−artifact

SubClassOf:

aspecto−fr:non−agentive−physical−object

Class: aspecto−fr:non−agentive−physical−object

SubClassOf:

dolce−lite−fr:physical−object

Class: aspecto−fr:rational−physical−object

SubClassOf:

aspecto−fr:agentive−physical−object

Class: dolce−lite−fr:amount−of−matter

Class: dolce−lite−fr:endurant

SubClassOf:

dolce−lite−fr:particular

Class: dolce−lite−fr:particular

Class: dolce−lite−fr:perdurant

SubClassOf:

dolce−lite−fr:particular

Class: dolce−lite−fr:physical−endurant

Class: dolce−lite−fr:physical−object

Class: dolce−lite−fr:quality

SubClassOf:

dolce−lite−fr:particular

Class: dolce−lite−fr:quality−space

Class: dolce−lite−fr:state

APPENDIX C

ASPECTO-FR-KB AND DOLCE-LITE-FR

C.1 Aspecto-FR-KB Ontology

Prefix: aspecto−fr: <http://www.ou.edu/coa/ontologies/2018/8/Aspecto−FR#>

Prefix: DOLCE−Lite−FR: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#>

Prefix: aspect−fr−kb: <http://www.ou.edu/coa/ontologies/2018/8/Aspecto−FR−KB#>

Prefix: owl: <http://www.w3.org/2002/07/owl#>

Prefix: rdf: <http://www.w3.org/1999/02/22−rdf−syntax−ns#>

Prefix: rdfs: <http://www.w3.org/2000/01/rdf−schema#>

Prefix: swrl: <http://www.w3.org/2003/11/swrl#>

Prefix: swrla: <http://swrl.stanford.edu/ontologies/3.3/swrla.owl#>

Prefix: swrlb: <http://www.w3.org/2003/11/swrlb#>

Prefix: xml: <http://www.w3.org/XML/1998/namespace>

Prefix: xsd: <http://www.w3.org/2001/XMLSchema#>

Ontology: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR−KB>

AnnotationProperty: rdfs:comment

AnnotationProperty: swrla:isRuleEnabled

Datatype: rdf:PlainLiteral

Datatype: xsd:boolean

ObjectProperty: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#
_AS_aspect_of_function>

ObjectProperty: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#_AS_environment>

ObjectProperty: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#_DF_artifact>

ObjectProperty: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#_DF_inPD>

ObjectProperty: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#_DF_outPD>

ObjectProperty: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#_EF_inPD>

Range:

<http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#InPD>

ObjectProperty: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#_EF_in_environment>

ObjectProperty: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#_EF_in_mode_of_deployment>

ObjectProperty: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#_EF_nominal_artifact>

ObjectProperty: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#_EF_outPD>

Range:

<http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#OutPD>

ObjectProperty: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#_MD_causal_model>

ObjectProperty: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#_MD_causal_model_OutPD>

ObjectProperty: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#_b_constrained_participant_in>

ObjectProperty: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#_c_caused_by>

Characteristics:

Transitive

ObjectProperty: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#_c_causes>

ObjectProperty: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#_d_caused_by>

ObjectProperty: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#_d_causes>

ObjectProperty: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#_n_causes>

ObjectProperty: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#b0>

ObjectProperty: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#b1>

ObjectProperty: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#
behavior_constraint_on_perdurant>

ObjectProperty: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#
behavioral_constraint_on>

ObjectProperty: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#causal_participant
>

ObjectProperty: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#
causal_participant_in>

ObjectProperty: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#
nominal_device_function>

ObjectProperty: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#
nominal_participant>

ObjectProperty: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#
nominal_participant_in>

ObjectProperty: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#r_behavior>

ObjectProperty: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#r_environment>

ObjectProperty: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#r_function>

ObjectProperty: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#r_inpd>

ObjectProperty: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#r_life>

ObjectProperty: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#r_outpd>

ObjectProperty: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR−KB#conflict>
SubPropertyChain:
<http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#_b_constrained_participant_in
> o <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR−KB#opposition> o
inverse (<http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#
_b_constrained_participant_in>)

ObjectProperty: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR−KB#conflict_high
>

ObjectProperty: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR−KB#conflict_low>

ObjectProperty: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR−KB#opposition>
Characteristics:

Symmetric

ObjectProperty: DOLCE−Lite−FR:life

ObjectProperty: DOLCE−Lite−FR:life−of

ObjectProperty: DOLCE−Lite−FR:part

ObjectProperty: DOLCE−Lite−FR:part−of

ObjectProperty: DOLCE−Lite−FR:participant

ObjectProperty: DOLCE−Lite−FR:participant−in

ObjectProperty: DOLCE−Lite−FR:proper−part

ObjectProperty: DOLCE−Lite−FR:proper−part−of

ObjectProperty: DOLCE−Lite−FR:weak−connection

DataProperty: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR−KB#
CT_true_realizability>

Range:

xsd:boolean

DataProperty: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR−KB#
DF_satisfiability>

Characteristics:

Functional

Range:

xsd:boolean

DataProperty: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR−KB#EF_realizability
>

Class: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#Actual_life>

Class: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#Behavior>

Class: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#EPD>

Class: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#Env1>

EquivalentTo:

(DOLCE−Lite−FR:weak−connection some <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#Environment>) or (DOLCE−Lite−FR:part−of value <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR−KB#Env1>)

Class: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#Environment>

Class: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#InPD>

Class: http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#OutPD

Class: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#Satisfiable_D_function>

Class: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#Technical_artifact>

Class: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#Technical_system>

Class: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#_Behavior>

Class: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#_D_function>

Class: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#_E_function>

Class: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#_P_causal_pattern>

Class: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#_Possible_behavior>

Class: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR−KB#Allow>

SubClassOf:

<http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR−KB#Primitive_behavior>

Class: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR−KB#Balance>

SubClassOf:

<http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR−KB#Primitive_behavior>

Class: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR−KB#Block>

SubClassOf:

<http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR−KB#Primitive_behavior>

Class: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR−KB#Decrease>

SubClassOf:

<http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR−KB#Primitive_behavior>

Class: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR−KB#Explosion>

SubClassOf:

DOLCE−Lite−FR:achievement

Class: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR−KB#Flowing>

SubClassOf:

DOLCE−Lite−FR:state

Class: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR−KB#Frictioning>

SubClassOf:

DOLCE−Lite−FR:state,

<http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#_c_caused_by> some
(<http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR−KB#Flowing> or <http
://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR−KB#Grinding> or <http
://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR−KB#Rotating> or <http
://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR−KB#Sliding>)

Class: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR−KB#Grinding>

SubClassOf:

DOLCE−Lite−FR:state

Class: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR−KB#Increase>

SubClassOf:

<http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR−KB#Primitive_behavior>

Class: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR−KB#Primitive_behavior>

SubClassOf:

<http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#Behavior>

Class: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR−KB#Rotating>

SubClassOf:

DOLCE−Lite−FR:state

Class: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR−KB#Sliding>

SubClassOf:

DOLCE−Lite−FR:state

Class: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR−KB#Vibrating>

SubClassOf:

DOLCE−Lite−FR:state,

<http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#_c_caused_by> some <http://
www.ou.edu/coa/ontologies/2018/6/Aspecto−FR−KB#Frictioning>,

<http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#_c_caused_by> only DOLCE−Lite−FR:state

Class: DOLCE−Lite−FR:achievement

Class: DOLCE−Lite−FR:state

Individual: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR−KB#Env1>

Rule:

<http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#b0>(?f, ?sf), <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#_P_causal_pattern>(?f), <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#b1>(?sf, ?b1), <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#Satisfiable_D_function>(?sf) −> <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#b0>(?f, ?b1)

Rule:

<http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#behavior_constraint_on_perdurant>(?b1, ?o), <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#_D_function>(?f), <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#behavior_constraint_on_perdurant>(?b0, ?i), <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#b0>(?f, ?b0), <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#b1>(?f, ?b1) −> <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#_DF_outPD>(?f, ?o), <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#_DF_inPD>(?f, ?i)

Rule:

<http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#behavior_constraint_on_perdurant>(?b1, ?o), <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#b1>(?ef, ?f1), <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#_E_function>(?ef), <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#b1>(?f1, ?b1), <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#_P_causal_pattern>(?f1) −> <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#b1>(?ef, ?b1), <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#_EF_outPD>(?ef, ?o)

Rule:

<http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#_E_function>(?ef), <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#_DF_inPD>(?ef, ?i) −> <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#_EF_inPD>(?ef, ?i)

Rule:

<http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#_E_function>(?ef), <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#_DF_outPD>(?ef, ?o) −> <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#_EF_outPD>(?ef, ?o)

Rule:

<http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#behavior_constraint_on_perdurant>(?b1, ?o), <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#b1>(?ef, ?b1), <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#b0>(?ef, ?b0), <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#_E_function>(?ef), <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#behavior_constraint_on_perdurant>(?b0, ?i) −> <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#_EF_outPD>(?ef, ?o), <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#_EF_inPD>(?ef, ?i)

Rule:

<http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#_c_caused_by>(?o, ?i), <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#_EF_inPD>(?d, ?i), <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#_EF_outPD>(?d, ?o) −> <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR−KB#EF_realizability>(?d, true)

C.2 DOLCE-Lite-FR Ontology

Prefix: dol: <http://www.loa−cnr.it/ontologies/DOLCE−Lite.owl#>

Prefix: owl: <http://www.w3.org/2002/07/owl#>

Prefix: rdf: <http://www.w3.org/1999/02/22−rdf−syntax−ns#>

Prefix: rdfs: <http://www.w3.org/2000/01/rdf−schema#>

Prefix: xml: <http://www.w3.org/XML/1998/namespace>

Prefix: xsd: <http://www.w3.org/2001/XMLSchema#>

Ontology: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl>

Annotations:

rdfs:comment "Modified by Andres Cavieres. Simplified structure with no axiomatization to improve inference performance.",

rdfs:comment "The DOLCE and DnS ontologies. OWL engineering by Aldo Gangemi."^^xsd: string,

owl: versionInfo "397

"^^xsd:string,

owl: versionInfo " classified

"^^xsd:string

AnnotationProperty: owl:versionInfo

AnnotationProperty: rdfs:comment

Datatype: xsd:string

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#abstract− location−of>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#exact−location−of>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#abstract−region>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#non−physical−endurant>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#abstract−location>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#abstract−location>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#exact−location>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#non−physical−endurant>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#abstract−region>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#abstract−location−of>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#atomic−part−of>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#part−of>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#atomic−part>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#atomic−part>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#part>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#atomic−part−of>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#boundary−of>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#proper−part−of>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#boundary>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#boundary>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#proper−part>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#boundary−of>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#constant−participant−in>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#participant−in>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#endurant>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#perdurant>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#constant−participant>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#constant−participant>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#participant>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#perdurant>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#endurant>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#constant−participant−in>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#exact−location−of>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#generic−location−of>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#region>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#exact−location>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#exact−location>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#generic−location>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#region>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#exact−location−of>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#generic−constituent−of>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#immediate−relation−i>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#generic−constituent>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#generic−constituent>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#immediate−relation>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#generic−constituent−of>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#generic−dependent>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#immediate−relation>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#generically−dependent−on
>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#generic−location
−of>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#mediated−relation−i>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#generic−location>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#generic−location
>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#mediated−relation>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#generic−location−of>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#generically−dependent−on>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#immediate−relation−i>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#generic−dependent>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#has−quale>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#q−location>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#quality>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#quale>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#quale−of>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#has−quality>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#immediate−relation−i>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#quality>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#inherent−in>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#has−t−quality>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#has−quality>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#quality>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#t−inherent−in>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#host−of>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#specific−constant−dependent>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#physical−endurant>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#feature>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#host>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#host>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#specifically−constantly−dependent−on>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#feature>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#physical−endurant>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#host−of>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#identity−c>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#immediate−relation>

Characteristics:

Transitive

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#identity−c>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#identity−n>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#immediate−relation>

Characteristics:

Transitive

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#identity−n>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#immediate−relation−i>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#immediate−relation>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#immediate−relation>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#immediate−relation−i>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#inherent−in>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#immediate−relation>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#quality>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#has−quality>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#life−of>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#constant−participant>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#perdurant>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#endurant>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#life>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#life>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#constant−participant−in>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#endurant>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#perdurant>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#life−of>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#mediated−relation−i>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#mediated−relation>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#mediated−relation>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#mediated−relation−i>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#mereologically−coincides>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#temporary−part>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#endurant>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#endurant>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#mereologically−coincides>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#overlaps>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#mediated−relation>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#overlaps>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#part−of>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#immediate−relation−i>

Characteristics:

Transitive ,

Reflexive

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#part>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#part>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#immediate−relation>

Characteristics:

Transitive ,

Reflexive

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#part−of>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#participant−in>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#immediate−relation−i>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#endurant>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#perdurant>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#participant>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#participant>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#immediate−relation>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#perdurant>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#endurant>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#participant−in>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#partly−compresent>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#mediated−relation>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#partly−compresent>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#physical−location−of>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#exact−location−of>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#physical−region>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#physical−endurant>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#physical−location>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#physical−location>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#exact−location>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#physical−endurant>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#physical−region>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#physical−location−of>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#proper−part−of>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#part−of>

Characteristics:

Transitive

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#proper−part>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#proper−part>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#part>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#proper−part−of>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#q−location−of>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#immediate−relation−i>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#region>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#quality>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#q−location>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#q−location>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#immediate−relation>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#quality>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#region>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#q−location−of>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#q−present−at>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#mediated−relation>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#physical−quality>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#time−interval>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#time−of−q−presence−of>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#quale−of>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#q−location−of>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#quale>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#quality>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#has−quale>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#r−location−of>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#immediate−relation−i>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#region>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#region>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#r−location>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#r−location>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#immediate−relation>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#region>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#region>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#r−location−of>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#sibling−part>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#mediated−relation>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#sibling−part>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#spatio−temporal
−presence−of>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#exact−location−of>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#spatio−temporal−region>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#spatio−temporally−present−at>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#spatio−temporally−present−at>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#exact−location>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#spatio−temporal−region>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#spatio−temporal−presence−of>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#specific−constant−constituent−of>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#immediate−relation−i>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#specific−constant−constituent>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#specific−constant−constituent>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#immediate−relation>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#specific−constant−constituent−of>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#specific−constant−dependent>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#immediate−relation>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#specifically−constantly−dependent−on>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#specifically−constantly−dependent−on>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#immediate−relation−i>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#specific−constant−dependent>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#strong−connection>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#mediated−relation>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#strong−connection>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#t−inherent−in>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#inherent−in>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#quality>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#has−t−quality>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#temporary−atomic−part−of>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#temporary−proper−part−of>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#endurant>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#endurant>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#temporary−atomic−part>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#temporary−atomic−part>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#temporary−proper−part>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#endurant>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#endurant>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#temporary−atomic−part−of>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#temporary−part−of>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#part−of>,

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#partly−compresent>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#endurant>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#endurant>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#temporary−part>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#temporary−part>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#part>,

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#partly−compresent>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#endurant>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#endurant>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#temporary−part−of>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#temporary−participant−in>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#participant−in>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#endurant>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#perdurant>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#temporary−participant>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#temporary−participant>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#participant>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#perdurant>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#endurant>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#temporary−participant−in
>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#temporary−
proper−part−of>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#proper−part−of>,

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#temporary−part−of>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#endurant>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#endurant>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#temporary−proper−part>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#temporary−
proper−part>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#proper−part>,

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#temporary−part>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#endurant>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#endurant>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#temporary−proper−part−of
>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#time−of−q−
presence−of>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#mediated−relation−i>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#time−interval>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#physical−quality>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#q−present−at>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#total−constant−
participant−in>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#constant−participant−in>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#endurant>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#perdurant>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#total−constant−participant
>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#total−constant−
participant>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#constant−participant>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#perdurant>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#endurant>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#total−constant−participant
−in>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#total−temporary
−participant−in>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#temporary−participant−in
>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#endurant>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#perdurant>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#total−temporary−
participant>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#total−temporary
−participant>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#temporary−participant>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#perdurant>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#endurant>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#total−temporary−
participant−in>

ObjectProperty: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#weak−connection>

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#immediate−relation>

Domain:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

Range:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

InverseOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#weak−connection>

Class: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#abstract−quality>

SubClassOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#quality>

DisjointWith:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#physical−quality>, <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#temporal−quality>

Class: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#abstract−region>

SubClassOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#region>

DisjointWith:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#physical−region>, <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#temporal−region>

Class: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#abstract>

SubClassOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

DisjointWith:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#endurant>, <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#perdurant>, <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#quality>

Class: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#accomplishment>

SubClassOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#event>

Class: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#achievement>

SubClassOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#event>

Class: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#amount−of−matter>

SubClassOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#physical−endurant>

DisjointWith:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#feature>, <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#physical−object>

Class: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#arbitrary−sum>

SubClassOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#endurant>,

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#part> some <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#endurant>

DisjointWith:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#non−physical−endurant>, <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#physical−endurant>

Class: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#dependent−place>

SubClassOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#feature>

Class: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#endurant>

DisjointWith:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#abstract>, <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#perdurant>, <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#quality>

Class: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#event>

SubClassOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#perdurant>

Class: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#feature>

SubClassOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#physical−endurant>,

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#host> some <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#physical−endurant>

DisjointWith:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#amount−of−matter>, <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#physical−object>

Class: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#non−physical−endurant>

SubClassOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#endurant>,

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#has−quality> only <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#abstract−quality>,

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#part> only <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#non−physical−endurant>

DisjointWith:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#arbitrary−sum>, <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#physical−endurant>

Class: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#non−physical−object>

SubClassOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#non−physical−endurant>,

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#generically−dependent−on > some <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#physical−endurant>,

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#part> only <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#non−physical−object>

Class: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#particular>

Class: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#perdurant>

DisjointWith:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#abstract>, <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#endurant>, <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#quality>

Class: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#physical−endurant>

SubClassOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#endurant>,

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#has−quality> some <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#physical−quality>,

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#has−quality> some <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#spatial−location_q>,

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#has−quality> only <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#physical−quality>,

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#part> only <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#physical−endurant>,

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#specific−constant−constituent> only <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#physical−endurant>

DisjointWith:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#arbitrary−sum>, <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#non−physical−endurant>

Class: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#physical−object>

SubClassOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#physical−endurant>

DisjointWith:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#amount−of−matter>, <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#feature>

Class: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#physical−quality>

SubClassOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#quality>

DisjointWith:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#abstract−quality>, <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#temporal−quality>

Class: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#physical−region>

SubClassOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#region>

DisjointWith:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#abstract−region>, <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#temporal−region>

Class: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#process>

SubClassOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#stative>

Class: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#proposition>

SubClassOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#abstract>

Class: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#quale>

SubClassOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#region>

Class: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#quality−space>

SubClassOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#region>

Class: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#quality>

DisjointWith:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#abstract>, <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#endurant>, <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#perdurant>

Class: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#region>

SubClassOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#abstract>

Class: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#relevant−part>

SubClassOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#feature>

Class: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#set>

SubClassOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#abstract>

Class: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#space−region>

SubClassOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#physical−region>

Class: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#spatial−location_q>

SubClassOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#physical−quality>

Class: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#spatio−temporal−region>

SubClassOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#space−region>

Class: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#state>

SubClassOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#stative>

Class: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#stative>

SubClassOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#perdurant>

Class: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#temporal−location_q>

SubClassOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#temporal−quality>

Class: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#temporal−quality>

SubClassOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#quality>

DisjointWith:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#abstract−quality>, <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#physical−quality>

Class: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#temporal−region>

SubClassOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#region>

DisjointWith:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#abstract−region>, <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#physical−region>

Class: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#time−interval>

SubClassOf:

<http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#temporal−region>

APPENDIX D

CASE STUDY 1: ELECTRONIC DEVICE

Prefix: dev: <http://www.ou.edu/coa/ontologies/2018/4/dev#>

Prefix: aspecto−fr: <http://www.ou.edu/coa/ontologies/2018/6/aspecto−fr#>

Prefix: DOLCE−Lite−FR: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#>

Prefix: owl: <http://www.w3.org/2002/07/owl#>

Prefix: rdf: <http://www.w3.org/1999/02/22−rdf−syntax−ns#>

Prefix: rdfs: <http://www.w3.org/2000/01/rdf−schema#>

Prefix: swrl: <http://www.w3.org/2003/11/swrl#>

Prefix: swrla: <http://swrl.stanford.edu/ontologies/3.3/swrla.owl#>

Prefix: swrlb: <http://www.w3.org/2003/11/swrlb#>

Prefix: xml: <http://www.w3.org/XML/1998/namespace>

Prefix: xsd: <http://www.w3.org/2001/XMLSchema#>

Ontology: <http://www.ou.edu/coa/ontologies/2018/4/dev>

Import: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR>

Import: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR−KB>

AnnotationProperty: aspecto−fr:_c_causes

AnnotationProperty: aspecto−fr:_d_causes

AnnotationProperty: rdfs:comment

AnnotationProperty: rdfs:label

AnnotationProperty: swrla:isRuleEnabled

Datatype: rdf:PlainLiteral

Datatype: xsd:boolean

Datatype: xsd:decimal

Datatype: xsd:integer

Datatype: xsd:string

ObjectProperty: dev:made_of

SubPropertyOf:

DOLCE−Lite−FR:specific−constant−constituent

ObjectProperty: dev:produces_part

Domain:

dev:Injection_Molding

Range:

aspecto−fr: Technical_artifact

and (dev:made_of some dev:Thermoplastic)

ObjectProperty: aspecto−fr:causal_participant_in_u

ObjectProperty: aspecto−fr:_c_causes

ObjectProperty: aspecto−fr:nominal_participant

ObjectProperty: aspecto−fr:nominal_participant_in

ObjectProperty: DOLCE−Lite−FR:part

ObjectProperty: DOLCE−Lite−FR:part−of

ObjectProperty: DOLCE−Lite−FR:participant−in

ObjectProperty: DOLCE−Lite−FR:proper−part

ObjectProperty: DOLCE−Lite−FR:proper−part−of

ObjectProperty: DOLCE−Lite−FR:specific−constant−constituent

DataProperty: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR−KB#
q_behavioral_constraint>

DataProperty: dev:has_area

Characteristics:

Functional

Domain:

DOLCE−Lite−FR:physical−object

Range:

xsd:decimal

DataProperty: dev:has_thickness

Range:

xsd:decimal

DataProperty: dev:has_thickness_constraint

DataProperty: dev:has_volume_capacity

DataProperty: dev:q_heat_output

Characteristics:

Functional

Domain:

dev:Electronic_subystem

Range:

xsd:decimal

SubPropertyOf:

<http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR−KB#q_behavioral_constraint>

Class: dev:Casing

SubClassOf:

aspecto−fr: Technical_artifact

Class: dev:Electronic_subystem

SubClassOf:

aspecto−fr:Technical_system

Class: dev:Injection_Molding

SubClassOf:

DOLCE−Lite−FR:process

Class: dev:Plastic

SubClassOf:

DOLCE−Lite−FR:amount−of−matter

Class: dev:Thermoplastic

SubClassOf:

dev:Plastic

Class: dev:Ventilation_opening

SubClassOf:

aspecto−fr:Technical_system

Class: dev: _All_Causal_Participants_in_Electrical_System_of_Device

EquivalentTo:

(DOLCE−Lite−FR:part some (aspecto−fr:causal_participant_in_u some (aspecto−fr:
_c_causes value dev:e1.2_outPD)))
and (DOLCE−Lite−FR:proper−part−of value <http://www.ou.edu/coa/ontologies
/2018/6/Aspecto−FR−KB#Env1>)

Class: dev:_Causal_Participants_in_Overall_Device_Function

EquivalentTo:

(DOLCE−Lite−FR:part some (aspecto−fr:causal_participant_in_u some (aspecto−fr:
_c_causes some ({dev:e1_outPD}))))

and (DOLCE−Lite−FR:proper−part−of value <http://www.ou.edu/coa/ontologies
/2018/6/Aspecto−FR−KB#Env1>)

Class: dev: _Direct_Causal_Participants_in_Electrical_System_of_Device

EquivalentTo:

(DOLCE−Lite−FR:part some (aspecto−fr:causal_participant_in_u value dev:e1.2_outPD))
and (DOLCE−Lite−FR:proper−part−of value <http://www.ou.edu/coa/ontologies
/2018/6/Aspecto−FR−KB#Env1>)

Class: dev:_Heat_Exhaust_Opening_Participation_in_Impact_Resistance

EquivalentTo:

(DOLCE−Lite−FR:part some (aspecto−fr:causal_participant_in_u some (aspecto−fr:
_c_causes value dev:e1.4_outPD)))
and (DOLCE−Lite−FR:proper−part−of value <http://www.ou.edu/coa/ontologies
/2018/6/Aspecto−FR−KB#Env1>)

Class: dev:_Nominal_Participant_in_Device_Device_Function

EquivalentTo:

(DOLCE−Lite−FR:part some (aspecto−fr:nominal_participant_in value dev:e1_outPD))
and (DOLCE−Lite−FR:proper−part−of value <http://www.ou.edu/coa/ontologies
/2018/6/Aspecto−FR−KB#Env1>)

Class: dev:_Nominal_Participants_in_Impact_Resistance

EquivalentTo:

(DOLCE−Lite−FR:part some (inverse (aspecto−fr:nominal_participant) value dev:e1.4
_outPD))
and (DOLCE−Lite−FR:proper−part−of value <http://www.ou.edu/coa/ontologies
/2018/6/Aspecto−FR−KB#Env1>)

Class: dev:q7

EquivalentTo:

aspecto−fr:Env1
and (DOLCE−Lite−FR:part−of some (aspecto−fr:nominal_participant_in some ({dev:
e5_outPD})))

Class: aspecto−fr:Env1

EquivalentTo:

DOLCE−Lite−FR:proper−part−of value <http://www.ou.edu/coa/ontologies/2018/6/
Aspecto−FR−KB#Env1>

Class: aspecto−fr: Technical_artifact

Class: aspecto−fr:Technical_system

Class: DOLCE−Lite−FR:amount−of−matter

Class: DOLCE−Lite−FR:physical−object

Class: DOLCE−Lite−FR:process

Individual: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR−KB#Env1>

Facts:

DOLCE−Lite−FR:proper−part dev:a1,

DOLCE−Lite−FR:proper−part dev:a6,

DOLCE−Lite−FR:proper−part dev:air

Individual: dev:a1

Annotations:

rdfs : label " electronic device"@en

Facts:

aspecto−fr:nominal_participant_in dev:e1_outPD,

DOLCE−Lite−FR:participant−in dev:e1.1,

DOLCE−Lite−FR:proper−part dev:a1.2,

DOLCE−Lite−FR:proper−part dev:a1.3,

DOLCE−Lite−FR:proper−part dev:a1.4

Individual: dev:a1.2

Annotations:

rdfs : label " electrical system"@en

Types:

dev:Electronic_subystem

Facts:

aspecto−fr:nominal_participant_in dev:e1.2_outPD,

dev:q_heat_output 10

Individual: dev:a1.3

Annotations:

rdfs : label " ventilation system"@en

Facts:

aspecto−fr:nominal_participant_in dev:e1.3_outPD,

DOLCE−Lite−FR:proper−part dev:a1.4.1

Individual: dev:a1.4

Annotations:

rdfs : label "casing"@en

Facts:

dev:made_of dev:a7,

aspecto−fr:nominal_participant_in dev:e1.4_outPD,

DOLCE−Lite−FR:participant−in dev:e5,

DOLCE−Lite−FR:proper−part dev:a1.4.1,

dev:has_thickness 12

Individual: dev:a1.4.1

Annotations:

rdfs:label "heat exhaust opening"@en

Types:

dev:Ventilation_opening

Facts:

aspecto−fr:nominal_participant_in dev:e1.3.4

Individual: dev:a6

Annotations:

rdfs:label "injection molding station"

Facts:

aspecto−fr:nominal_participant_in dev:e5,
not dev:has_thickness_constraint 15.5

Individual: dev:a7

Types:

dev:Thermoplastic

Individual: dev:air

Annotations:

rdfs:comment "air"

Facts:

DOLCE−Lite−FR:participant−in dev:e1.3.4

Individual: dev:e1.1

Annotations:

rdfs:label "Device processes",

aspecto−fr:_d_causes <http://www.ou.edu/coa/ontologies/2018/4/dev#e1.2>

Facts:

DOLCE−Lite−FR:proper−part dev:e1.2_outPD

Individual: dev:e1.2

Annotations:

aspecto−fr:_c_causes <http://www.ou.edu/coa/ontologies/2018/4/dev#e1.2.1>,

rdfs:comment "life of a1.2"

Facts:

DOLCE−Lite−FR:proper−part dev:e1.2.1,

DOLCE−Lite−FR:proper−part dev:e1.2_outPD

Individual: dev:e1.2.1

Annotations:

rdfs:comment "electronic sub−processes",

aspecto−fr:_d_causes <http://www.ou.edu/coa/ontologies/2018/4/dev#e1.2_outPD>

Individual: dev:e1.2_outPD

Annotations:

rdfs:label "electronic sub−function"@en,

aspecto−fr:_d_causes <http://www.ou.edu/coa/ontologies/2018/4/dev#e1_outPD>,

aspecto−fr:_c_causes <http://www.ou.edu/coa/ontologies/2018/4/dev#e1.3.1>

Individual: dev:e1.3

Annotations:

rdfs:label "heat transfer"@en

Facts:

DOLCE−Lite−FR:proper−part dev:e1.3.1,

DOLCE−Lite−FR:proper−part dev:e1.3.2,

DOLCE−Lite−FR:proper−part dev:e1.3.3,

DOLCE−Lite−FR:proper−part dev:e1.3_outPD

Individual: dev:e1.3.1

Annotations:

rdfs:comment "radiant heat",

aspecto−fr:_c_causes <http://www.ou.edu/coa/ontologies/2018/4/dev#e1.3.2>

Individual: dev:e1.3.2

Annotations:

aspecto−fr:_c_causes <http://www.ou.edu/coa/ontologies/2018/4/dev#e1.3_outPD>,

rdfs:comment "convective heat",

aspecto−fr:_c_causes <http://www.ou.edu/coa/ontologies/2018/4/dev#e1.3.3>

Individual: dev:e1.3.3

Annotations:

rdfs:label "heat containment"@en,

aspecto−fr:_c_causes <http://www.ou.edu/coa/ontologies/2018/4/dev#e1.2.1>

Individual: dev:e1.3.4

Annotations:

rdfs:comment "air flow for heat release",

aspecto−fr:_d_causes <http://www.ou.edu/coa/ontologies/2018/4/dev#e1.3_outPD>

Individual: dev:e1.3_outPD

Annotations:

rdfs:label "ventilation sub−function"@en

Individual: dev:e1.4

Facts:

DOLCE−Lite−FR:proper−part dev:e1.4.2,

DOLCE−Lite−FR:proper−part dev:e1.4.3,

DOLCE−Lite−FR:proper−part dev:e1.4_outPD

Individual: dev:e1.4.1

Annotations:

aspecto−fr:_c_causes <http://www.ou.edu/coa/ontologies/2018/4/dev#e1.4.2>,

rdfs:label "loading / impact"@en

Individual: dev:e1.4.2

Annotations:

rdfs:label "load distribution"@en,

aspecto−fr:_d_causes <http://www.ou.edu/coa/ontologies/2018/4/dev#e1.4.3>

Individual: dev:e1.4.3

Annotations:

rdfs:label "impact absortion"@en,

aspecto−fr:_d_causes <http://www.ou.edu/coa/ontologies/2018/4/dev#e1.4_outPD>

Individual: dev:e1.4_outPD

Annotations:

rdfs:label "structural sub−function"@en

Facts:

DOLCE−Lite−FR:proper−part dev:e1.4.3

Individual: dev:e1_inPD

Annotations:

aspecto−fr:_d_causes <http://www.ou.edu/coa/ontologies/2018/4/dev#e1.1>,

rdfs:label "device inputs"@en

Individual: dev:e1_outPD

Annotations:

rdfs:label "device main function"@en

Individual: dev:e5

Annotations:

rdfs:label "injection molding process",

aspecto−fr:_d_causes <http://www.ou.edu/coa/ontologies/2018/4/dev#e5_outPD>

Facts:

DOLCE−Lite−FR:proper−part dev:e5_InPD,

DOLCE−Lite−FR:proper−part dev:e5_outPD

Individual: dev:e5_InPD

Annotations:

aspecto−fr:_d_causes <http://www.ou.edu/coa/ontologies/2018/4/dev#e5>

Individual: dev:e5_outPD

Annotations:

rdfs:label "injection molding production"

Individual: dev:life−of−device

Facts:

DOLCE−Lite−FR:proper−part dev:e1.1,

DOLCE−Lite−FR:proper−part dev:e1.2,

DOLCE−Lite−FR:proper−part dev:e1.3,

DOLCE−Lite−FR:proper−part dev:e1.4.3

Rule:

swrlb:multiply(?h, ?r, 1.75), dev:Ventilation_opening(?v), dev:Electronic_subystem(?s), dev:q_heat_output(?s, ?h) −> dev:has_area(?v, ?r)

Rule:

swrlb:greaterThan(?r, 30), dev:Ventilation_opening(?v), dev:has_area(?v, ?r) −> aspecto−fr:causal_participant_in_u(dev:a1.2, dev:e1.4.2), aspecto−fr:causal_participant_in_u(?v, dev:e1.4.2)

Rule:

dev:Ventilation_opening(?v), swrlb:lessThan(?r, 20), dev:has_area(?v, ?r) −> aspecto−fr:causal_participant_in_u(?v, dev:e1.2.1), aspecto−fr:causal_participant_in_u(dev:air, dev:e1.2.1)

Rule:

dev:has_thickness(dev:a1.4, ?x), swrlb:greaterThan(?x, 15) −> aspecto−fr:causal_participant_in_u(dev:a1.4, dev:e1.3.3)

APPENDIX E

CASE STUDIES 2-3: PV RACKING SYSTEM

Prefix: Aspect−FR−Solar: <http://www.ou.edu/coa/ontologies/2018/4/Aspect−FR−Solar#>

Prefix: aspecto−fr: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#>

Prefix: DOLCE−Lite−FR: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#>

Prefix: owl: <http://www.w3.org/2002/07/owl#>

Prefix: rdf: <http://www.w3.org/1999/02/22−rdf−syntax−ns#>

Prefix: rdfs: <http://www.w3.org/2000/01/rdf−schema#>

Prefix: swrl: <http://www.w3.org/2003/11/swrl#>

Prefix: swrla: <http://swrl.stanford.edu/ontologies/3.3/swrla.owl#>

Prefix: swrlb: <http://www.w3.org/2003/11/swrlb#>

Prefix: xml: <http://www.w3.org/XML/1998/namespace>

Prefix: xsd: <http://www.w3.org/2001/XMLSchema#>

Ontology: <http://www.ou.edu/coa/ontologies/2018/4/Aspect−FR−Solar>

Import: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR>

Import: <http://www.ou.edu/coa/ontologies/2018/6/Aspect−FR−KB>

AnnotationProperty: rdfs:comment

AnnotationProperty: rdfs:label

Datatype: rdf:PlainLiteral

Datatype: xsd:boolean

Datatype: xsd:decimal

Datatype: xsd:int

Datatype: xsd:integer

Datatype: xsd:string

ObjectProperty: Aspect−FR−Solar:component_of

SubPropertyOf:

DOLCE−Lite−FR:temporary−proper−part−o

ObjectProperty: aspecto−fr:_AS_aspect_of_function

ObjectProperty: aspecto−fr:_DF_artifact

SubPropertyChain:

aspecto−fr:b0 o aspecto−fr:_DF_artifac

ObjectProperty: aspecto−fr:_DF_inPD

ObjectProperty: aspecto−fr:_DF_outPD

ObjectProperty: aspecto−fr:_EF_inPD

ObjectProperty: aspecto−fr:_EF_in_environment

ObjectProperty: aspecto−fr:_EF_in_mode_of_deployment

ObjectProperty: aspecto−fr:_EF_nominal_artifact

ObjectProperty: aspecto−fr:_EF_outPD

ObjectProperty: aspecto−fr:_b_has_capability

ObjectProperty: aspecto−fr:_c_caused_by

ObjectProperty: aspecto−fr:_c_causes

ObjectProperty: aspecto−fr:_d_caused_by

ObjectProperty: aspecto−fr:_d_causes

ObjectProperty: aspecto−fr:b0

ObjectProperty: aspecto−fr:b1

Annotations:

rdfs:comment "Roof assembly"

ObjectProperty: aspecto−fr:behavior_constraint_on_perdurant

ObjectProperty: aspecto−fr:behavioral_constraint_on

ObjectProperty: aspecto−fr:has_component

ObjectProperty: aspecto−fr:nominal_participant_in

SubPropertyChain:

inverse (aspecto−fr:_EF_nominal_artifact) o aspecto−fr:_DF_outPD

ObjectProperty: aspecto−fr:r_artifact

ObjectProperty: aspecto−fr:s_adjacent_to

Characteristics:

Symmetric

ObjectProperty: aspecto−fr:s_connected_to

ObjectProperty: aspecto−fr:s_directly_connected_to

Characteristics:

Symmetric

ObjectProperty: DOLCE−Lite−FR:part

ObjectProperty: DOLCE−Lite−FR:part−of

ObjectProperty: DOLCE−Lite−FR:participant−in

ObjectProperty: DOLCE−Lite−FR:proper−part

Characteristics:

Transitive

ObjectProperty: DOLCE−Lite−FR:proper−part−of

ObjectProperty: DOLCE−Lite−FR:temporary−proper−part−of

ObjectProperty: DOLCE−Lite−FR:weak−connection

DataProperty: Aspect−FR−Solar:has_UV_degradation_property

SubPropertyOf:

Aspect−FR−Solar:has_environmental_property

DataProperty: Aspect−FR−Solar:has_aerodynamic_coefficient

Annotations:

rdfs : label "Cd"

SubPropertyOf:

Aspect−FR−Solar:has_mechanical_property

DataProperty: Aspect−FR−Solar:has_anti−corrosion_property

Annotations:

rdfs : label ”anti−corrosion treatment”

SubPropertyOf:

Aspect−FR−Solar:has_environmental_property

DataProperty: Aspect−FR−Solar:has_area

Characteristics:

Functional

Range:

xsd:decimal

SubPropertyOf:

Aspect−FR−Solar:has_structural_property

DataProperty: Aspect−FR−Solar:has_area_constrain

SubPropertyOf:

Aspect−FR−Solar:has_structural_constraint

DataProperty: Aspect−FR−Solar:has_behavioral_constraint

SubPropertyOf:

owl:topDataProperty

DataProperty: Aspect−FR−Solar:has_behavioral_property

Annotations:

rdfs : comment ”n”ˆˆxsd:string

SubPropertyOf:

owl:topDataProperty

DataProperty: Aspect−FR−Solar:has_bending_moment

Annotations:

rdfs : label ”lbf−ft”

SubPropertyOf:

Aspect−FR−Solar:has_mechanical_property

DataProperty: Aspect−FR−Solar:has_business_constrain

Range:

xsd:decimal

SubPropertyOf:

owl:topDataProperty

DataProperty: Aspect−FR−Solar:has_business_property

Range:

xsd:decimal

SubPropertyOf:

owl:topDataProperty

DataProperty: Aspect−FR−Solar:has_charging_time

Annotations:

rdfs : label "minutes"

SubPropertyOf:

Aspect−FR−Solar:has_electrical_property

DataProperty: Aspect−FR−Solar:has_compression_capabilit

Annotations:

rdfs : label "psf"

SubPropertyOf:

Aspect−FR−Solar:has_mechanical_property

DataProperty: Aspect−FR−Solar:has_conductor_gauge

Annotations:

rdfs : label "wire gauge"

SubPropertyOf:

Aspect−FR−Solar:has_electrical_property

DataProperty: Aspect−FR−Solar:has_density

SubPropertyOf:

Aspect−FR−Solar:has_structural_property

DataProperty: Aspect−FR−Solar:has_density_constraint

SubPropertyOf:

Aspect−FR−Solar:has_structural_constraint

DataProperty: Aspect−FR−Solar:has_electrical_constrain

SubPropertyOf:

Aspect−FR−Solar:has_behavioral_constraint

DataProperty: Aspect−FR−Solar:has_electrical_property

SubPropertyOf:

Aspect−FR−Solar:has_behavioral_property

DataProperty: Aspect−FR−Solar:has_electrical_resistitivity

Annotations:

rdfs : label "milliohms"

SubPropertyOf:

Aspect−FR−Solar:has_electrical_property

DataProperty: Aspect−FR−Solar:has_environmental_constraint

SubPropertyOf:

Aspect−FR−Solar:has_behavioral_constraint

DataProperty: Aspect−FR−Solar:has_environmental_property

SubPropertyOf:

Aspect−FR−Solar:has_behavioral_property

DataProperty: Aspect−FR−Solar:has_ergonomic_advantage

Annotations:

rdfs : label "yes/no"

SubPropertyOf:

Aspect−FR−Solar:has_installation_property

DataProperty: Aspect−FR−Solar:has_ergonomic_constraint

SubPropertyOf:

Aspect−FR−Solar:has_behavioral_constraint

DataProperty: Aspect−FR−Solar:has_extension

Annotations:

rdfs : label "m"

SubPropertyOf:

Aspect−FR−Solar:has_electrical_property

DataProperty: Aspect−FR−Solar:has_feature

DataProperty: Aspect−FR−Solar:has_friction_coefficient

Annotations:

rdfs : label "uN"

SubPropertyOf:

Aspect−FR−Solar:has_mechanical_property

DataProperty: Aspect−FR−Solar:has_galvanization

Annotations:

rdfs : label "galvanization"

SubPropertyOf:

Aspect−FR−Solar:has_environmental_property

DataProperty: Aspect−FR−Solar:has_installation_advantage

Annotations:

rdfs : label "description"

SubPropertyOf:

Aspect−FR−Solar:has_installation_property

DataProperty: Aspect−FR−Solar:has_installation_constraint

Range:

xsd:int

SubPropertyOf:

Aspect−FR−Solar:has_behavioral_constraint

DataProperty: Aspect−FR−Solar:has_installation_cost

SubPropertyOf:

Aspect−FR−Solar:has_business_property

DataProperty: Aspect−FR−Solar:has_installation_cost_constraint

SubPropertyOf:

Aspect−FR−Solar:has_business_constraint

DataProperty: Aspect−FR−Solar:has_installation_property

SubPropertyOf:

Aspect−FR−Solar:has_behavioral_property

DataProperty: Aspect−FR−Solar:has_installation_speed

Annotations:

rdfs:label " Install speed"

SubPropertyOf:

Aspect−FR−Solar:has_installation_property

DataProperty: Aspect−FR−Solar:has_installation_steps

Annotations:

rdfs:label " Install steps"

Characteristics:

Functional

Range:

xsd:int

SubPropertyOf:

Aspect−FR−Solar:has_installation_property

DataProperty: Aspect−FR−Solar:has_installation_steps_sum

Range:

xsd:int

SubPropertyOf:

Aspect−FR−Solar:has_installation_property

DataProperty: Aspect−FR−Solar:has_installation_time

Annotations:

rdfs:label "seconds"

Characteristics:

Functional

Range:

xsd:int

SubPropertyOf:

Aspect−FR−Solar:has_installation_property

DataProperty: Aspect−FR−Solar:has_installation_time_sum

Range:

xsd:int

SubPropertyOf:

Aspect−FR−Solar:has_installation_property

DataProperty: Aspect−FR−Solar:has_length

SubPropertyOf:

Aspect−FR−Solar:has_structural_property

DataProperty: Aspect−FR−Solar:has_length_constraint

SubPropertyOf:

Aspect−FR−Solar:has_structural_constraint

DataProperty: Aspect−FR−Solar:has_manufacturing_cost

Annotations:

rdfs : label "Dollars"

Characteristics:

Functional

Range:

xsd:decimal

SubPropertyOf:

Aspect−FR−Solar:has_business_property

DataProperty: Aspect−FR−Solar:has_manufacturing_cost_constraint

SubPropertyOf:

Aspect−FR−Solar:has_business_constraint

DataProperty: Aspect−FR−Solar:has_manufacturing_cost_sum

Range:

xsd:decimal

SubPropertyOf:

Aspect−FR−Solar:has_business_property

DataProperty: Aspect−FR−Solar:has_mass

SubPropertyOf:

Aspect−FR−Solar:has_structural_property

DataProperty: Aspect−FR−Solar:has_mass_constraint

SubPropertyOf:

Aspect−FR−Solar:has_structural_constraint

DataProperty: Aspect−FR−Solar:has_max_amps

Annotations:

rdfs : label "amps"

SubPropertyOf:

Aspect−FR−Solar:has_electrical_property

DataProperty: Aspect−FR−Solar:has_mechanical_constraint

SubPropertyOf:

Aspect−FR−Solar:has_behavioral_constraint

DataProperty: Aspect−FR−Solar:has_mechanical_property

SubPropertyOf:

Aspect−FR−Solar:has_behavioral_property

DataProperty: Aspect−FR−Solar:has_power_supply

Annotations:

rdfs : label "V"

SubPropertyOf:

Aspect−FR−Solar:has_electrical_property

DataProperty: Aspect−FR−Solar:has_protective_pad

Characteristics:

Functional

Range:

xsd:boolean

DataProperty: Aspect−FR−Solar:has_shipping_cost

SubPropertyOf:

Aspect−FR−Solar:has_business_property

DataProperty: Aspect−FR−Solar:has_shipping_cost_constraint

SubPropertyOf:

Aspect−FR−Solar:has_business_constraint

DataProperty: Aspect−FR−Solar:has_structural_constraint

Range:

xsd:decimal

SubPropertyOf:

owl:topDataProperty

DataProperty: Aspect−FR−Solar:has_structural_property

Range:

xsd:decimal

SubPropertyOf:

owl:topDataProperty

DataProperty: Aspect−FR−Solar:has_supply_cycles

Annotations:

rdfs : label "supply cycles"

SubPropertyOf:

Aspect−FR−Solar:has_electrical_property

DataProperty: Aspect−FR−Solar:has_tensile_capability

Annotations:

rdfs : label "psf"

SubPropertyOf:

Aspect−FR−Solar:has_mechanical_property

DataProperty: Aspect−FR−Solar:has_thermal_expansion_coefficient

Annotations:

rdfs : label "F x10^−6"

SubPropertyOf:

Aspect−FR−Solar:has_mechanical_property

DataProperty: Aspect−FR−Solar:has_thickness

Range:

xsd:decimal

SubPropertyOf:

Aspect−FR−Solar:has_structural_property

DataProperty: Aspect−FR−Solar:has_thickness_constraint

SubPropertyOf:

Aspect−FR−Solar:has_structural_constraint

DataProperty: Aspect−FR−Solar:has_torque

Annotations:

rdfs : label "N.m"

Range:

xsd:decimal

SubPropertyOf:

Aspect−FR−Solar:has_mechanical_property

DataProperty: Aspect−FR−Solar:has_torque_control

Annotations:

rdfs : label "yes/no"

Range:

xsd:boolean

SubPropertyOf:

Aspect−FR−Solar:has_mechanical_property

DataProperty: Aspect−FR−Solar:has_torsion_capability

SubPropertyOf:

Aspect−FR−Solar:has_mechanical_property

DataProperty: Aspect−FR−Solar:has_training_cost

SubPropertyOf:

Aspect−FR−Solar:has_business_property

DataProperty: Aspect−FR−Solar:has_training_cost_constraint

SubPropertyOf:

Aspect−FR−Solar:has_business_constraint

DataProperty: Aspect−FR−Solar:has_voltage_required

Annotations:

rdfs : label "V"

SubPropertyOf:

Aspect−FR−Solar:has_electrical_property

DataProperty: Aspect−FR−Solar:has_volume

SubPropertyOf:

Aspect−FR−Solar:has_structural_property

DataProperty: Aspect−FR−Solar:has_volume_constraint

SubPropertyOf:

Aspect−FR−Solar:has_structural_constraint

DataProperty: Aspect−FR−Solar:has_weight

Annotations:

rdfs : label "lb"

SubPropertyOf:

Aspect−FR−Solar:has_mechanical_property

DataProperty: Aspect−FR−Solar:has_weight_per_area

Annotations:

rdfs : label "psf"

SubPropertyOf:

Aspect−FR−Solar:has_mechanical_property

DataProperty: Aspect−FR−Solar:has_wind_speed_limit

Annotations:

rdfs : label ”mph”

SubPropertyOf:

Aspect−FR−Solar:has_mechanical_property

DataProperty: Aspect−FR−Solar:has_wire_management

Annotations:

rdfs : label ” level of integration”

SubPropertyOf:

Aspect−FR−Solar:has_installation_property

DataProperty: owl:topDataProperty

Class: <http://www.ou.edu/coa/ontologies/2018/6/Aspect−FR−KB#Increase>

Class: <http://www.ou.edu/coa/ontologies/2018/6/Aspect−FR−KB#Sliding>

Class: Aspect−FR−Solar:Alignment

SubClassOf:

DOLCE−Lite−FR:process

Class: Aspect−FR−Solar:Aspect_system_electrical_installation

Class: Aspect−FR−Solar:Aspect_system_full_installation

Class: Aspect−FR−Solar:Aspect_system_grounding_function

Class: Aspect−FR−Solar:Aspect_system_maintain_form

EquivalentTo:

(DOLCE−Lite−FR:part some (DOLCE−Lite−FR:participant−in some (aspecto−fr:_c_causes some (inverse (aspecto−fr:_EF_outPD) value Aspect−FR−Solar:_fe7.2_maintain_form))))

and (DOLCE−Lite−FR:proper−part−of value Aspect−FR−Solar:Env2)

Class: Aspect−FR−Solar:Aspect_system_maintain_position

EquivalentTo:

(DOLCE−Lite−FR:part some (DOLCE−Lite−FR:participant−in some (aspecto−fr:_c_causes some (inverse (aspecto−fr:_DF_outPD) value Aspect−FR−Solar:_fe7.1_maintain_position))))

and (DOLCE−Lite−FR:proper−part−of value Aspect−FR−Solar:Env2)

Class: Aspect−FR−Solar:Aspect_system_squaring_function

Class: Aspect−FR−Solar:Aspect_system_structural_function

EquivalentTo:

(DOLCE−Lite−FR:part some (DOLCE−Lite−FR:participant−in some (aspecto−fr:
_c_causes some (inverse (aspecto−fr:_DF_outPD) value Aspect−FR−Solar:
_fe7_maintain_structural_integrity))))
and (DOLCE−Lite−FR:proper−part−of value Aspect−FR−Solar:Env2)

Class: Aspect−FR−Solar:Aspect_system_wire_management

EquivalentTo:

(DOLCE−Lite−FR:part some (DOLCE−Lite−FR:participant−in some (aspecto−fr:
_c_causes some (inverse (aspecto−fr:_DF_outPD) value Aspect−FR−Solar:_fe5.1
_wire_management))))
and (DOLCE−Lite−FR:proper−part−of value <http://www.ou.edu/coa/ontologies
/2018/6/Aspect−FR−KB#Env1>)

Class: Aspect−FR−Solar:Attachment

SubClassOf:

DOLCE−Lite−FR:process

Class: Aspect−FR−Solar:Ballast

SubClassOf:

Aspect−FR−Solar:Racking_component,
aspecto−fr:s_directly_connected_to some Aspect−FR−Solar:Ballast_tray,
DOLCE−Lite−FR:participant−in value Aspect−FR−Solar:e_gravity

Class: Aspect−FR−Solar:Ballast_tray

SubClassOf:

Aspect−FR−Solar:Racking_component,

inverse (aspecto−fr:_DF_artifact) value Aspect−FR−Solar:_fd4.1_resist_drag_by_friction,

inverse (aspecto−fr:_DF_artifact) value Aspect−FR−Solar:_fd4.1_resist_uplift_by_ballast

Class: Aspect−FR−Solar:Ballast_tray_pad

SubClassOf:

Aspect−FR−Solar:Racking_component,

aspecto−fr:s_directly_connected_to some Aspect−FR−Solar:Roof_membrane,

aspecto−fr:_b_has_capability value Aspect−FR−Solar:b_reduces_drag_by_friction

Class: Aspect−FR−Solar:Base_rail

SubClassOf:

Aspect−FR−Solar:Racking_component,

aspecto−fr:s_directly_connected_to some Aspect−FR−Solar:Ballast_tray,

aspecto−fr:_b_has_capability value Aspect−FR−Solar:b_thermal_expansion_coefficient,

DOLCE−Lite−FR:participant−in value Aspect−FR−Solar:e2.2b_pre−squaring,

DOLCE−Lite−FR:participant−in value Aspect−FR−Solar:e_electrical_conductivity

Class: Aspect−FR−Solar:Bolt

SubClassOf:

Aspect−FR−Solar:Fastener

Class: Aspect−FR−Solar:Bonding

SubClassOf:

DOLCE−Lite−FR:state

Class: Aspect−FR−Solar:Bonding_capability

SubClassOf:

aspecto−fr:Behavior

Class: Aspect−FR−Solar:Bonding_washer

SubClassOf:

Aspect−FR−Solar:Washer

Class: Aspect−FR−Solar:Bracing

SubClassOf:

DOLCE−Lite−FR:state

Class: Aspect−FR−Solar:Bracing_E

EquivalentTo:

Aspect−FR−Solar:Bracing
and (inverse (aspecto−fr:_DF_outPD) some (aspecto−fr:_DF_artifact some
((aspecto−fr:_b_has_capability some Aspect−FR−Solar:Bracing_capability)
and (aspecto−fr:s_connected_to some (DOLCE−Lite−FR:part−of value <http://
www.ou.edu/coa/ontologies/2018/6/Aspect−FR−KB#Env1>))

and (DOLCE−Lite−FR:part−of some Aspect−FR−Solar:PV_racking_assembly))))

SubClassOf:

aspecto−fr:CauseMD,

aspecto−fr:_c_causes value Aspect−FR−Solar:e7.1_maintain_form

Class: Aspect−FR−Solar:Bracing_capability

SubClassOf:

aspecto−fr:Behavior

Class: Aspect−FR−Solar:Bracing_device

EquivalentTo:

(aspecto−fr:has_component min 1 Aspect−FR−Solar:Moment_resistant_connection)
and (aspecto−fr:s_connected_to min 2 (aspecto−fr:s_adjacent_to some Aspect−FR−Solar:
Mounting_base_top))

SubClassOf:

inverse (aspecto−fr:_DF_artifact) value Aspect−FR−Solar:_fd4.2_bracing

Class: Aspect−FR−Solar:Bracket

SubClassOf:

Aspect−FR−Solar:Connection

Class: Aspect−FR−Solar:Buckling

SubClassOf:

DOLCE−Lite−FR:state

Class: Aspect−FR−Solar:Building

SubClassOf:

aspecto−fr:Technical_system

Class: Aspect−FR−Solar:Building_component

SubClassOf:

aspecto−fr: Technical_artifact

Class: Aspect−FR−Solar:Clip

SubClassOf:

Aspect−FR−Solar:Fastener

Class: Aspect−FR−Solar:Connection

SubClassOf:

Aspect−FR−Solar:Racking_component

Class: Aspect−FR−Solar:Constraint

SubClassOf:

Aspect−FR−Solar:Requirement

Class: Aspect−FR−Solar:Deflected_deck_bay

Class: Aspect−FR−Solar:Deflection_E

SubClassOf:

aspecto−fr:CauseMD

Class: Aspect−FR−Solar:Drag_resistance_capability

SubClassOf:

aspecto−fr:Behavior

Class: Aspect−FR−Solar:Dragging

SubClassOf:

DOLCE−Lite−FR:state

Class: Aspect−FR−Solar:Dragging_E

EquivalentTo:

<http://www.ou.edu/coa/ontologies/2018/6/Aspect−FR−KB#Sliding>
and (inverse (aspecto−fr:_DF_outPD) some (aspecto−fr:_DF_artifact some
((aspecto−fr:_b_has_capability some Aspect−FR−Solar:Drag_resistance_capability)
and (aspecto−fr:s_connected_to some (DOLCE−Lite−FR:part−of value <http://
www.ou.edu/coa/ontologies/2018/6/Aspect−FR−KB#Env1>))
and (DOLCE−Lite−FR:part−of some Aspect−FR−Solar:PV_racking_assembly))))

SubClassOf:

aspecto−fr:CauseMD,

aspecto−fr:_d_causes value Aspect−FR−Solar:e7.1_maintain_position

Class: Aspect−FR−Solar:Electrical_component

SubClassOf:

Aspect−FR−Solar:Racking_component,

DOLCE−Lite−FR:participant−in value Aspect−FR−Solar:e3.3_eletrical_testing

Class: Aspect−FR−Solar:Electrical_system

SubClassOf:

aspecto−fr:Technical_system

Class: Aspect−FR−Solar:Empty_assembly_component

SubClassOf:

Aspect−FR−Solar:Racking_component

Class: Aspect−FR−Solar:Env2

EquivalentTo:

(DOLCE−Lite−FR:weak−connection some aspecto−fr:Environment) or (DOLCE−Lite−FR:part−of value Aspect−FR−Solar:Env2)

SubClassOf:

aspecto−fr:Environment

Class: Aspect−FR−Solar:Fastened

SubClassOf:

DOLCE−Lite−FR:achievement

DisjointWith:

Aspect−FR−Solar:Squared

Class: Aspect−FR−Solar:Fastener

SubClassOf:

Aspect−FR−Solar:Connection

Class: Aspect−FR−Solar:Fastening_E

EquivalentTo:

Aspect−FR−Solar:Fastened
and (inverse (aspecto−fr:_DF_outPD) some (aspecto−fr:_DF_artifact some
((aspecto−fr:_b_has_capability some Aspect−FR−Solar:Fastening_capability)
and (aspecto−fr:s_connected_to some (DOLCE−Lite−FR:part−of value <http://
www.ou.edu/coa/ontologies/2018/6/Aspect−FR−KB#Env1>)))))

SubClassOf:

aspecto−fr:CauseMD

Class: Aspect−FR−Solar:Fastening_capability

SubClassOf:

aspecto−fr:Behavior

Class: Aspect−FR−Solar:Grounding

SubClassOf:

DOLCE−Lite−FR:state

Class: Aspect−FR−Solar:Grounding_E

EquivalentTo:

Aspect−FR−Solar:Bonding

and (inverse (aspecto−fr:_DF_outPD) some (aspecto−fr:_DF_artifact some

(((aspecto−fr:s_connected_to some (DOLCE−Lite−FR:part−of value <http://www.

ou.edu/coa/ontologies/2018/6/Aspect−FR−KB#Env1>))

and (DOLCE−Lite−FR:part−of some Aspect−FR−Solar:PV_racking_assembly))

and (aspecto−fr:_b_has_capability some Aspect−FR−Solar:Bonding_capability))))

SubClassOf:

aspecto−fr:CauseMD,

aspecto−fr:_c_causes value Aspect−FR−Solar:e_electrical_conductivity

Class: Aspect−FR−Solar:Hinge_capability

SubClassOf:

aspecto−fr:Behavior

Class: Aspect−FR−Solar:Maintain_form

SubClassOf:

Aspect−FR−Solar:Maintain_structural_integrity

Class: Aspect−FR−Solar:Maintain_position

SubClassOf:

Aspect−FR−Solar:Maintain_structural_integrity

Class: Aspect−FR−Solar:Maintain_structural_integrity

SubClassOf:

DOLCE−Lite−FR:state

Class: Aspect−FR−Solar:Maintaining_form_and_position

EquivalentTo:

(Aspect−FR−Solar:Maintain_form or Aspect−FR−Solar:Maintain_position)
and ((aspecto−fr:_c_caused_by some Aspect−FR−Solar:Bracing_E) or (aspecto−fr:
_c_caused_by some Aspect−FR−Solar:Dragging_E) or (aspecto−fr:_c_caused_by some
Aspect−FR−Solar:Uplifting_E))

SubClassOf:

aspecto−fr:CauseMD,

aspecto−fr:_d_causes value Aspect−FR−Solar:e7_maintain_form_and_position

Class: Aspect−FR−Solar:Maintaining_structural_integrity

EquivalentTo:

aspecto−fr:_d_caused_by min 2 Aspect−FR−Solar:Maintaining_form_and_position

SubClassOf:

aspecto−fr:CauseMD,

aspecto−fr:_c_causes value Aspect−FR−Solar:e7_final

Class: Aspect−FR−Solar:Moment_resistant_capability

SubClassOf:

aspecto−fr:Behavior

Class: Aspect−FR−Solar:Moment_resistant_connection

EquivalentTo:

(aspecto−fr:has_component some Aspect−FR−Solar:Welded_connection) or (aspecto−fr:has_component min 2 Aspect−FR−Solar:Fastener)

SubClassOf:

Aspect−FR−Solar:Pin_connection,

aspecto−fr:_b_has_capability value Aspect−FR−Solar:b_moment_resistence_capability

Class: Aspect−FR−Solar:Mounting_base_bottom

SubClassOf:

Aspect−FR−Solar:Racking_component

Class: Aspect−FR−Solar:Mounting_base_top

SubClassOf:

Aspect−FR−Solar:Racking_component

Class: Aspect−FR−Solar:Mounting_clip_bottom

SubClassOf:

Aspect−FR−Solar:Racking_component

Class: Aspect−FR−Solar:Mounting_clip_top

SubClassOf:

Aspect−FR−Solar:Racking_component

Class: Aspect−FR−Solar:Over−deflected_bay

EquivalentTo:

Aspect−FR−Solar:Roof_deck_bay
and (DOLCE−Lite−FR:weak−connection some (DOLCE−Lite−FR:weak−connection
some (Aspect−FR−Solar:has_weight some xsd:decimal[> 30])))

Class: Aspect−FR−Solar:PV_installer

SubClassOf:

aspecto−fr:rational−physical−object

Class: Aspect−FR−Solar:PV_module

SubClassOf:

Aspect−FR−Solar:Electrical_component

Class: Aspect−FR−Solar:PV_racking_assembly

SubClassOf:

aspecto−fr:Technical_system

Class: Aspect−FR−Solar:PV_system

SubClassOf:

aspecto−fr:Technical_system

Class: Aspect−FR−Solar:Pin_connection

SubClassOf:

Aspect−FR−Solar:Bracket,

aspecto−fr:_b_has_capability value Aspect−FR−Solar:b_moment_transfer_capability

Class: Aspect−FR−Solar:Racking_component

SubClassOf:

aspecto−fr: Technical_artifact

Class: Aspect−FR−Solar:Requirement

SubClassOf:

DOLCE−Lite−FR:non−physical−object

Class: Aspect−FR−Solar:Rivet

SubClassOf:

Aspect−FR−Solar:Fastener

Class: Aspect−FR−Solar:Roof_deck

SubClassOf:

Aspect−FR−Solar:Building_component

Class: Aspect−FR−Solar:Roof_deck_bay

SubClassOf:

Aspect−FR−Solar:Building_component

Class: Aspect−FR−Solar:Roof_joist

SubClassOf:

Aspect−FR−Solar:Building_component

Class: Aspect−FR−Solar:Roof_membrane

SubClassOf:

Aspect−FR−Solar:Building_component,

aspecto−fr:s_connected_to some Aspect−FR−Solar:Roof_substrate

Class: Aspect−FR−Solar:Roof_structure

SubClassOf:

aspecto−fr:Technical_system

Class: Aspect−FR−Solar:Roof_substrate

SubClassOf:

Aspect−FR−Solar:Building_component,

aspecto−fr:s_connected_to some Aspect−FR−Solar:Roof_structure

Class: Aspect−FR−Solar:Screw

SubClassOf:

Aspect−FR−Solar:Fastener,

aspecto−fr:nominal_participant_in value Aspect−FR−Solar:e_screwing_2,

DOLCE−Lite−FR:participant−in value Aspect−FR−Solar:e_screwing_3

Class: Aspect−FR−Solar:Screw_driver

SubClassOf:

Aspect−FR−Solar:Tool

Class: Aspect−FR−Solar:Screwing_E

EquivalentTo:

Aspect−FR−Solar:Fastened

and (inverse (aspecto−fr:_DF_outPD) some (aspecto−fr:_DF_artifact some

((aspecto−fr:s_connected_to some (DOLCE−Lite−FR:part−of value <http://www.ou

.edu/coa/ontologies/2018/6/Aspect−FR−KB#Env1>))

and (aspecto−fr:_b_has_capability value Aspect−FR−Solar:b_screwing))))

SubClassOf:

Aspect−FR−Solar:Fastening_E,

aspecto−fr:_c_causes value Aspect−FR−Solar:e_piercing

Class: Aspect−FR−Solar:Slotting_E

EquivalentTo:

Aspect−FR−Solar:Fastened

and (inverse (aspecto−fr:_DF_outPD) some (aspecto−fr:_DF_artifact some

((aspecto−fr:s_connected_to some (DOLCE−Lite−FR:part−of value <http://www.ou

.edu/coa/ontologies/2018/6/Aspect−FR−KB#Env1>))

and (aspecto−fr:_b_has_capability value Aspect−FR−Solar:b_quick_fastening))))

SubClassOf:

Aspect−FR−Solar:Fastening_E,

aspecto−fr:_c_causes value Aspect−FR−Solar:e2.2a_pre−attachment

Class: Aspect−FR−Solar:Spacer

SubClassOf:

Aspect−FR−Solar:Connection

Class: Aspect−FR−Solar:Squared

SubClassOf:

DOLCE−Lite−FR:achievement

DisjointWith:

Aspect−FR−Solar:Fastened

Class: Aspect−FR−Solar:Squaring_E

EquivalentTo:

Aspect−FR−Solar:Squared
and (inverse (aspecto−fr:_DF_outPD) some (aspecto−fr:_EF_nominal_artifact some
((aspecto−fr:_b_has_capability some Aspect−FR−Solar:Squaring_capability)
and (DOLCE−Lite−FR:part some (aspecto−fr:s_connected_to min 2 Aspect−FR−
Solar:Mounting_base_top))
and (DOLCE−Lite−FR:part−of some Aspect−FR−Solar:PV_racking_assembly))))

SubClassOf:

aspecto−fr:CauseMD,

Class: Aspect−FR−Solar:Squaring_capability

SubClassOf:

aspecto−fr:Behavior

Class: Aspect−FR−Solar:Tool

SubClassOf:

aspecto−fr: Technical_artifact

Class: Aspect−FR−Solar:Unpacked

SubClassOf:

DOLCE−Lite−FR:achievement

Class: Aspect−FR−Solar:Unpacking_E

EquivalentTo:

Aspect−FR−Solar:Unpacked
and (inverse (aspecto−fr:_DF_outPD) some (aspecto−fr:_DF_artifact some (DOLCE−
Lite−FR:part some
((aspecto−fr:_b_has_capability some aspecto−fr:Behavior)
and (DOLCE−Lite−FR:proper−part−of some Aspect−FR−Solar:
PV_racking_assembly)))))

SubClassOf:

aspecto−fr:CauseMD,
aspecto−fr:_d_causes value Aspect−FR−Solar:e1.1_transporting

Class: Aspect−FR−Solar:Uplift_resistance_capability

SubClassOf:

aspecto−fr:Behavior

Class: Aspect−FR−Solar:Uplifting

SubClassOf:

DOLCE−Lite−FR:state

Class: Aspect−FR−Solar:Uplifting_E

EquivalentTo:

Aspect−FR−Solar:Uplifting
and (inverse (aspecto−fr:_DF_outPD) some (aspecto−fr:_DF_artifact some
((aspecto−fr:_b_has_capability some Aspect−FR−Solar:Uplift_resistance_capability)
and (aspecto−fr:s_connected_to some (DOLCE−Lite−FR:part−of value <http://
www.ou.edu/coa/ontologies/2018/6/Aspect−FR−KB#Env1>))
and (DOLCE−Lite−FR:part−of some Aspect−FR−Solar:PV_racking_assembly))))

SubClassOf:

aspecto−fr:CauseMD,
aspecto−fr:_d_causes value Aspect−FR−Solar:e7.1_maintain_position

Class: Aspect−FR−Solar:Washer

SubClassOf:

Aspect−FR−Solar:Spacer

Class: Aspect−FR−Solar:Welded_connection

SubClassOf:

Aspect−FR−Solar:Connection

Class: Aspect−FR−Solar:Wind_deflector

SubClassOf:

Aspect−FR−Solar:Racking_component,

aspecto−fr:s_connected_to some Aspect−FR−Solar:Mounting_base_top,

DOLCE−Lite−FR:participant−in value Aspect−FR−Solar:e4.2_drag,

DOLCE−Lite−FR:proper−part value Aspect−FR−Solar:a3.1_slot_bracket_set,

inverse (aspecto−fr:_DF_artifact) value Aspect−FR−Solar:_fd4.1 _resist_uplift_by_deflection

Class: Aspect−FR−Solar:Wired

SubClassOf:

DOLCE−Lite−FR:achievement

Class: Aspect−FR−Solar:Wiring_E

EquivalentTo:

Aspect−FR−Solar:Wired

and (inverse (aspecto−fr:_DF_outPD) some (aspecto−fr:_DF_artifact some ((aspecto−fr:_b_has_capability some Aspect−FR−Solar:Wiring_capability) and (DOLCE−Lite−FR:part some (aspecto−fr:s_connected_to some (DOLCE−Lite−FR:part−of value <http://www.ou.edu/coa/ontologies/2018/6/Aspect−FR−KB#Env1>)))

and (DOLCE−Lite−FR:part−of some Aspect−FR−Solar:PV_racking_assembly))))

SubClassOf:

Aspect−FR−Solar:Wired,

aspecto−fr:CauseMD,

aspecto−fr:_d_causes value Aspect−FR−Solar:e3.2_wire_management

Class: Aspect−FR−Solar:Wiring_capability

SubClassOf:

aspecto−fr:Behavior

Class: aspecto−fr:Behavior

Class: aspecto−fr:CauseMD

SubClassOf:

inverse (aspecto−fr:_DF_outPD) some (aspecto−fr:_DF_artifact some (DOLCE−Lite−FR:part some (aspecto−fr:_b_has_capability some aspecto−fr:Behavior)))

Class: aspecto−fr:Environment

Class: aspecto−fr:Realizable_E_Function

Class: aspecto−fr: Satisfiable_E_function

Class: aspecto−fr: Technical_artifact

Class: aspecto−fr:Technical_system

Class: aspecto−fr:_E_function

Class: aspecto−fr:material−artifact

SubClassOf:

aspecto−fr: r_artifact some Self

Class: aspecto−fr:rational−physical−object

Class: DOLCE−Lite−FR:achievement

Class: DOLCE−Lite−FR:non−physical−object

Class: DOLCE−Lite−FR:process

Class: DOLCE−Lite−FR:state

Individual: <http://www.ou.edu/coa/ontologies/2018/6/Aspect−FR−KB#Env1>

Facts:

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:a0_electrical_crew,

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:a0_hardware_crew,

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:a1_pv_racking,

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:a8_installation_tools,

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:a8_power_outlet,

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:a9_building

Individual: Aspect−FR−Solar:Env2

Facts:

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:a9_building

Individual: Aspect−FR−Solar:_as0

Facts:

aspecto−fr:_AS_aspect_of_function Aspect−FR−Solar:_fe0_flat_packing

Individual: Aspect−FR−Solar:_as1

Facts:

aspecto−fr:_AS_aspect_of_function Aspect−FR−Solar:_fe1_whole_installation

Individual: Aspect−FR−Solar:_as2

Facts:

aspecto−fr:_AS_aspect_of_function Aspect−FR−Solar:_fe2_hardware_installation

Individual: Aspect−FR−Solar:_as2.1

Facts:

aspecto−fr:_AS_aspect_of_function Aspect−FR−Solar:_fe2.1_squaring_function

Individual: Aspect−FR−Solar:_as2.2

Facts:

aspecto−fr:_AS_aspect_of_function Aspect−FR−Solar:_fe2.2_pv_module_attachment

Individual: Aspect−FR−Solar:_as3

Facts:

aspecto−fr:_AS_aspect_of_function Aspect−FR−Solar:_fe5_electrical_installation

Individual: Aspect−FR−Solar:_as4

Facts:

aspecto−fr:_AS_aspect_of_function Aspect−FR−Solar:_fe7.1_maintain_position

Individual: Aspect−FR−Solar:_as5

Facts:

aspecto−fr:_AS_aspect_of_function Aspect−FR−Solar:_fe5.2_electrical_grounding

Individual: Aspect−FR−Solar:_fd1_flat_packing_D_function

Facts:

aspecto−fr:_DF_artifact Aspect−FR−Solar:a3_wind_deflector,

aspecto−fr:b0 Aspect−FR−Solar:b_flat_packing,

aspecto−fr:b1 Aspect−FR−Solar:b_flat_unpacking

Individual: Aspect−FR−Solar:_fd2.1_screw_fastening_D_function

Facts:

aspecto−fr:_DF_artifact Aspect−FR−Solar:a3.1_screw_bracket_left,

aspecto−fr:b0 Aspect−FR−Solar:b_screwing_1,

aspecto−fr:b1 Aspect−FR−Solar:b_screwing

Individual: Aspect−FR−Solar:_fd2.1_slot_fastening_D_function

Facts:

aspecto−fr:_DF_artifact Aspect−FR−Solar:a3.1_slot_bracket_left,

aspecto−fr:_DF_artifact Aspect−FR−Solar:a3.1_slot_bracket_right,

aspecto−fr:b0 Aspect−FR−Solar:b_slotting_1,

aspecto−fr:b1 Aspect−FR−Solar:b_quick_fastening

Individual: Aspect−FR−Solar:_fd2_pre−squaring_D_function

Facts:

aspecto−fr:_DF_artifact Aspect−FR−Solar:a3_wind_deflector,

aspecto−fr:b0 Aspect−FR−Solar:_fd2.1_slot_fastening_D_function,

aspecto−fr:b1 Aspect−FR−Solar:b_integrated_squaring

Individual: Aspect−FR−Solar:_fd2_squaring_by_measure_D_function

Facts:

aspecto−fr:_DF_artifact Aspect−FR−Solar:a7.1_tape_measure,

aspecto−fr:_DF_artifact Aspect−FR−Solar:a7.2_chalk_line,

aspecto−fr:b0 Aspect−FR−Solar:b_measuring,

aspecto−fr:b1 Aspect−FR−Solar:b_normal_squaring

Individual: Aspect−FR−Solar:_fd3_wire_management_D_function

Facts:

aspecto−fr:_DF_artifact Aspect−FR−Solar:a3_wind_deflector,

aspecto−fr:b0 Aspect−FR−Solar:b_wiring,

aspecto−fr:b1 Aspect−FR−Solar:b_integrated_wire_bundling

Individual: Aspect−FR−Solar:_fd4.1_resist_drag_by_friction

Facts:

aspecto−fr:_DF_inPD Aspect−FR−Solar:e_loading_seismic,

aspecto−fr:_DF_inPD Aspect−FR−Solar:e_loading_wind,

aspecto−fr:b1 Aspect−FR−Solar:b_reduces_drag_by_friction

Individual: Aspect−FR−Solar:_fd4.1_resist_uplift_by_ballast

Facts:

aspecto−fr:b0 Aspect−FR−Solar:b_loading_wind,

aspecto−fr:b1 Aspect−FR−Solar:b_reduces_uplift_by_weight

Individual: Aspect−FR−Solar:_fd4.1_resist_uplift_by_deflection

Facts:

aspecto−fr:b0 Aspect−FR−Solar:b_loading_wind,

aspecto−fr:b1 Aspect−FR−Solar:b_keep_air_pressure_balance,

aspecto−fr:b1 Aspect−FR−Solar:b_reduces_uplift_by_deflection

Individual: Aspect−FR−Solar:_fd4.2_bracing

Facts:

aspecto−fr:b0 Aspect−FR−Solar:b_loading_seismic,

aspecto−fr:b0 Aspect−FR−Solar:b_loading_wind,

aspecto−fr:b1 Aspect−FR−Solar:b_bracing_capability

Individual: Aspect−FR−Solar:_fd5_electrical_bonding

Facts:

aspecto−fr:_DF_artifact Aspect−FR−Solar:a3.3_star_washer_left,

aspecto−fr:_DF_artifact Aspect−FR−Solar:a3.3_star_washer_right,

aspecto−fr:b0 Aspect−FR−Solar:b_tighted_by_torque,

aspecto−fr:b1 Aspect−FR−Solar:b_bonding_by_torque

Individual: Aspect−FR−Solar:_fe0_flat_packing

Annotations:

rdfs:comment "unpacking"@en

Types:

aspecto−fr:_E_function

Facts:

aspecto−fr:_EF_in_environment <http://www.ou.edu/coa/ontologies/2018/6/Aspect−FR−KB#Env1>,

aspecto−fr:_EF_in_mode_of_deployment Aspect−FR−Solar:_md1,

aspecto−fr:_EF_nominal_artifact Aspect−FR−Solar:a1_pv_racking,

aspecto−fr:b0 Aspect−FR−Solar:_fd1_flat_packing_D_function,

aspecto−fr:b1 Aspect−FR−Solar:b_flat_unpacking

Individual: Aspect−FR−Solar:_fe1_whole_installation

Annotations:

rdfs:comment "Maintain position"@en

Types:

aspecto−fr:_E_function

Facts:

aspecto−fr:_EF_in_environment <http://www.ou.edu/coa/ontologies/2018/6/Aspect−FR−KB#Env1>,

aspecto−fr:_EF_in_mode_of_deployment Aspect−FR−Solar:_md7,

aspecto−fr:_EF_nominal_artifact Aspect−FR−Solar:a1_pv_racking,

aspecto−fr:b0 Aspect−FR−Solar:b_staged,

aspecto−fr:b1 Aspect−FR−Solar:b_electrical_tested,

aspecto−fr:b1 Aspect−FR−Solar:b_hardware_installed

Individual: Aspect−FR−Solar:_fe2.1_squaring_function

Types:

aspecto−fr:_E_function

Facts:

aspecto−fr:_EF_in_environment <http://www.ou.edu/coa/ontologies/2018/6/Aspect−FR−KB#Env1>,

aspecto−fr:_EF_in_mode_of_deployment Aspect−FR−Solar:_md2.1,

aspecto−fr:_EF_nominal_artifact Aspect−FR−Solar:a1_pv_racking,

aspecto−fr:b0 Aspect−FR−Solar:_fd2_pre−squaring_D_function,

aspecto−fr:b1 Aspect−FR−Solar:b_squared

Individual: Aspect−FR−Solar:_fe2.2_pv_module_attachment

Annotations:

rdfs:comment "PV module attachment"@en

Types:

aspecto−fr:_E_function

Facts:

aspecto−fr:_EF_in_environment <http://www.ou.edu/coa/ontologies/2018/6/Aspect−FR−KB#Env1>,

aspecto−fr:_EF_in_mode_of_deployment Aspect−FR−Solar:_md2.2,

aspecto−fr:_EF_nominal_artifact Aspect−FR−Solar:a1_pv_racking,

aspecto−fr:b0 Aspect−FR−Solar:b_ballasted,

aspecto−fr:b1 Aspect−FR−Solar:b_pv_module_attachment

Individual: Aspect−FR−Solar:_fe2_hardware_installation

Annotations:

rdfs:comment "hardware installation"@en

Types:

aspecto−fr:_E_function

Facts:

aspecto−fr:_EF_in_environment <http://www.ou.edu/coa/ontologies/2018/6/Aspect−FR−KB#Env1>,

aspecto−fr:_EF_in_mode_of_deployment Aspect−FR−Solar:_md2,

aspecto−fr:_EF_nominal_artifact Aspect−FR−Solar:a1_pv_racking,

aspecto−fr:b0 Aspect−FR−Solar:b_flat_unpacking,

aspecto−fr:b1 Aspect−FR−Solar:b_hardware_installed

Individual: Aspect−FR−Solar:_fe5.1_wire_management

Annotations:

rdfs:comment "Wire management"@en

Types:

aspecto−fr:_E_function

Facts:

aspecto−fr:_EF_in_environment <http://www.ou.edu/coa/ontologies/2018/6/Aspect−FR−KB#Env1>,

aspecto−fr:_EF_in_mode_of_deployment Aspect−FR−Solar:_md3,

aspecto−fr:_EF_nominal_artifact Aspect−FR−Solar:a1_pv_racking,

aspecto−fr:b0 Aspect−FR−Solar:_fd3_wire_management_D_function,

aspecto−fr:b1 Aspect−FR−Solar:b_integrated_wire_management

Individual: Aspect−FR−Solar:_fe5.2_electrical_grounding

Types:

aspecto−fr:_E_function

Facts:

aspecto−fr:_EF_in_environment <http://www.ou.edu/coa/ontologies/2018/6/Aspect−FR−KB#Env1>,

aspecto−fr:_EF_in_mode_of_deployment Aspect−FR−Solar:_md5,

aspecto−fr:_EF_nominal_artifact Aspect−FR−Solar:a1_pv_racking,

aspecto−fr:b0 Aspect−FR−Solar:_fd5_electrical_bonding,

aspecto−fr:b1 Aspect−FR−Solar:b_grounding

Individual: Aspect−FR−Solar:_fe5_electrical_installation

Annotations:

rdfs:comment "electrical installation"@en

Types:

aspecto−fr:_E_function

Facts:

aspecto−fr:_EF_in_environment <http://www.ou.edu/coa/ontologies/2018/6/Aspect−FR−KB#Env1>,

aspecto−fr:_EF_in_mode_of_deployment Aspect−FR−Solar:_md3,

aspecto−fr:_EF_nominal_artifact Aspect−FR−Solar:a1_pv_racking,

aspecto−fr:b0 Aspect−FR−Solar:b_wiring,

aspecto−fr:b1 Aspect−FR−Solar:b_electrical_tested

Individual: Aspect−FR−Solar:_fe7.1_maintain_position

Types:

aspecto−fr:_E_function

Facts:

aspecto−fr:_EF_in_environment Aspect−FR−Solar:Env2,

aspecto−fr:_EF_in_mode_of_deployment Aspect−FR−Solar:_md7,

aspecto−fr:_EF_nominal_artifact Aspect−FR−Solar:a1_pv_racking,

aspecto−fr:b0 Aspect−FR−Solar:_fd4.1_resist_drag_by_friction,

aspecto−fr:b0 Aspect−FR−Solar:_fd4.1_resist_uplift_by_ballast ,

aspecto−fr:b0 Aspect−FR−Solar:_fd4.1_resist_uplift_by_deflection ,

aspecto−fr:b1 Aspect−FR−Solar:b_maintain_position_capability

Individual: Aspect−FR−Solar:_fe7.2_maintain_form

Types:

aspecto−fr:_E_function

Facts:

aspecto−fr:_EF_in_environment Aspect−FR−Solar:Env2,

aspecto−fr:_EF_in_mode_of_deployment Aspect−FR−Solar:_md7,

aspecto−fr:_EF_nominal_artifact Aspect−FR−Solar:a1_pv_racking,

aspecto−fr:b0 Aspect−FR−Solar:_fd4.2_bracing,

aspecto−fr:b1 Aspect−FR−Solar:b_maintain_form_capability

Individual: Aspect−FR−Solar:_fe7_maintain_structural_integrity

Types:

aspecto−fr:_E_function

Facts:

aspecto−fr:_EF_in_environment Aspect−FR−Solar:Env2,

aspecto−fr:_EF_in_mode_of_deployment Aspect−FR−Solar:_md7,

aspecto−fr:_EF_nominal_artifact Aspect−FR−Solar:a1_pv_racking,

aspecto−fr:_EF_nominal_artifact Aspect−FR−Solar:a9.1_roof_assembly,

aspecto−fr:b0 Aspect−FR−Solar:b_maintain_form_capability,

aspecto−fr:b0 Aspect−FR−Solar:b_maintain_position_capability,

aspecto−fr:b1 Aspect−FR−Solar:b_maintain_s_integrity_capability

Individual: Aspect−FR−Solar:_fe_global

Annotations:

rdfs:comment "Global installation function"@en

Facts:

aspecto−fr:_EF_in_environment <http://www.ou.edu/coa/ontologies/2018/6/Aspect−FR−KB#Env1>

Individual: Aspect−FR−Solar:_md1

Individual: Aspect−FR−Solar:_md2

Individual: Aspect−FR−Solar:_md2.1

Annotations:

rdfs:comment "Manufacturing costs"@en

Individual: Aspect−FR−Solar:_md2.2

Individual: Aspect−FR−Solar:_md3

Individual: Aspect−FR−Solar:_md5

Individual: Aspect−FR−Solar:_md7

Individual: Aspect−FR−Solar:a0_electrical_crew

Facts:

DOLCE−Lite−FR:participant−in Aspect−FR−Solar:e3.1_wiring,

DOLCE−Lite−FR:participant−in Aspect−FR−Solar:e_electrical_grounding

Individual: Aspect−FR−Solar:a0_hardware_crew

Facts:

DOLCE−Lite−FR:participant−in Aspect−FR−Solar:e1_staging,

DOLCE−Lite−FR:participant−in Aspect−FR−Solar:e2.2b_pre−squaring,

DOLCE−Lite−FR:participant−in Aspect−FR−Solar:e2.3a_squared

Individual: Aspect−FR−Solar:a1_pv_racking

Annotations:

rdfs : label "PV system"@en

Types:

Aspect−FR−Solar:PV_racking_assembly

Facts:

aspecto−fr:_b_has_capability Aspect−FR−Solar:b_anti_corrosion_capability,

aspecto−fr:_b_has_capability Aspect−FR−Solar:b_grounding,

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:a2_base_rail_left,

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:a2_base_rail_right,

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:a3_wind_deflector,

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:a4_ballast_tray_left,

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:a4_ballast_tray_right,

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:a5_inverter,

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:a5_junction_box,

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:a5_wires,

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:a6_pv_module,

Aspect−FR−Solar:has_manufacturing_cost 0

Individual: Aspect−FR−Solar:a2.1_mounting_base_left

Annotations:

rdfs:comment "Mounting base top"@en

Types:

Aspect−FR−Solar:Mounting_base_top

Facts:

aspecto−fr:s_adjacent_to Aspect−FR−Solar:a2.1_mounting_base_right,

aspecto−fr:s_directly_connected_to Aspect−FR−Solar:a2_base_rail_left,

Aspect−FR−Solar:has_manufacturing_cost 1.152

DifferentFrom:

Aspect−FR−Solar:a2.1_mounting_base_right

Individual: Aspect−FR−Solar:a2.1_mounting_base_right

Annotations:

rdfs:comment "Mounting base top"@en

Types:

Aspect−FR−Solar:Mounting_base_top

Facts:

aspecto−fr:s_directly_connected_to Aspect−FR−Solar:a2_base_rail_right,
Aspect−FR−Solar:has_manufacturing_cost 1.152

DifferentFrom:

Aspect−FR−Solar:a2.1_mounting_base_left

Individual: Aspect−FR−Solar:a2.2_bottom_mounting_base_left

Annotations:

rdfs:comment "Mounting base bottom"@en

Types:

Aspect−FR−Solar:Mounting_base_bottom

Facts:

aspecto−fr:s_directly_connected_to Aspect−FR−Solar:a2_base_rail_left,
Aspect−FR−Solar:has_manufacturing_cost 0.857

Individual: Aspect−FR−Solar:a2.2_bottom_mounting_base_right

Annotations:

rdfs:comment "Mounting base bottom"@en

Types:

Aspect−FR−Solar:Mounting_base_bottom

Facts:

Aspect−FR−Solar:has_manufacturing_cost 0.857

Individual: Aspect−FR−Solar:a2_base_rail_left

Annotations:

rdfs:label "Base rail left"@en

Types:

Aspect−FR−Solar:Base_rail

Facts:

aspecto−fr:s_directly_connected_to Aspect−FR−Solar:a4_ballast_tray_left,

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:a2.1_mounting_base_left,

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:a2.2_bottom_mounting_base_left,

Aspect−FR−Solar:has_manufacturing_cost 1.63

Individual: Aspect−FR−Solar:a2_base_rail_right

Annotations:

rdfs:comment "Base rail right"

Types:

Aspect−FR−Solar:Base_rail

Facts:

aspecto−fr:s_directly_connected_to Aspect−FR−Solar:a4_ballast_tray_right,

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:a2.1_mounting_base_right,

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:a2.2_bottom_mounting_base_right,

Aspect−FR−Solar:has_manufacturing_cost 1.63

Individual: Aspect−FR−Solar:a3.1_screw_bracket_left

Annotations:

rdfs:comment "Wind deflector left connector"@en

Types:

Aspect−FR−Solar:Bracket

Facts:

aspecto−fr:has_component Aspect−FR−Solar:a3.2a_screw_1,

aspecto−fr:has_component Aspect−FR−Solar:a3.2a_screw_2,

aspecto−fr:s_directly_connected_to Aspect−FR−Solar:a2.1_mounting_base_left,

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:a3.3_star_washer_left

DifferentFrom:

Aspect−FR−Solar:a3.1_screw_bracket_right

Individual: Aspect−FR−Solar:a3.1_screw_bracket_right

Facts:

aspecto−fr:has_component Aspect−FR−Solar:a3.2b_screw_1,

aspecto−fr:has_component Aspect−FR−Solar:a3.2b_screw_2,

aspecto−fr:s_directly_connected_to Aspect−FR−Solar:a2.1_mounting_base_right,

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:a3.3_star_washer_right

DifferentFrom:

Aspect−FR−Solar:a3.1_screw_bracket_left

Individual: Aspect−FR−Solar:a3.1_screw_bracket_set

Facts:

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:a3.1_screw_bracket_left,
DOLCE−Lite−FR:proper−part Aspect−FR−Solar:a3.1_screw_bracket_right,
DOLCE−Lite−FR:proper−part Aspect−FR−Solar:a3.2a_screw_1,
DOLCE−Lite−FR:proper−part Aspect−FR−Solar:a3.2b_screw_1

Individual: Aspect−FR−Solar:a3.1_slot_bracket_left

Annotations:

rdfs:comment "Wind deflector right connector"

Types:

Aspect−FR−Solar:Pin_connection

Facts:

aspecto−fr:s_directly_connected_to Aspect−FR−Solar:a2.1_mounting_base_left,
DOLCE−Lite−FR:participant−in Aspect−FR−Solar:e_electrical_continuity,
Aspect−FR−Solar:has_manufacturing_cost 0.1

DifferentFrom:

Aspect−FR−Solar:a3.1_slot_bracket_right

Individual: Aspect−FR−Solar:a3.1_slot_bracket_right

Types:

Aspect−FR−Solar:Pin_connection

Facts:

aspecto−fr:s_directly_connected_to Aspect−FR−Solar:a2.1_mounting_base_right,

DOLCE−Lite−FR:participant−in Aspect−FR−Solar:e_electrical_continuity

DifferentFrom:

Aspect−FR−Solar:a3.1_slot_bracket_left

Individual: Aspect−FR−Solar:a3.1_slot_bracket_set

Facts:

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:a3.1_slot_bracket_left,

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:a3.1_slot_bracket_right

Individual: Aspect−FR−Solar:a3.2a_screw_1

Types:

Aspect−FR−Solar:Screw

DifferentFrom:

Aspect−FR−Solar:a3.2b_screw_1

Individual: Aspect−FR−Solar:a3.2a_screw_2

Types:

Aspect−FR−Solar:Screw

DifferentFrom:

Aspect−FR−Solar:a3.2b_screw_1

Individual: Aspect−FR−Solar:a3.2b_screw_1

Types:

Aspect−FR−Solar:Screw

DifferentFrom:

Aspect−FR−Solar:a3.2a_screw_1,

Aspect−FR−Solar:a3.2a_screw_2,

Aspect−FR−Solar:a3.2b_screw_2

Individual: Aspect−FR−Solar:a3.2b_screw_2

Types:

Aspect−FR−Solar:Screw

DifferentFrom:

Aspect−FR−Solar:a3.2b_screw_1

Individual: Aspect−FR−Solar:a3.3_star_washer_left

Types:

Aspect−FR−Solar:Bonding_washer

Facts:

aspecto−fr:s_connected_to Aspect−FR−Solar:a2.1_mounting_base_left

Individual: Aspect−FR−Solar:a3.3_star_washer_right

Types:

Aspect−FR−Solar:Bonding_washer

Facts:

aspecto−fr:s_connected_to Aspect−FR−Solar:a2.1_mounting_base_right

Individual: Aspect−FR−Solar:a3_wind_deflector

Annotations:

rdfs:comment "The two connector parts of the wind deflector exemplify the use of the functional property 's_directly_connected_to'. The functional restriction on the property does not allow the wind deflector to be directly connected more than once. Hence, two connection parts were created, each handling a 's_directly_connected_to' property to a mounting rail.",

rdfs:label "Wind deflector"@en

Types:

Aspect−FR−Solar:Wind_deflector

Facts:

aspecto−fr:has_component Aspect−FR−Solar:a3.1_screw_bracket_left,

aspecto−fr:has_component Aspect−FR−Solar:a3.1_screw_bracket_right,

aspecto−fr:has_component Aspect−FR−Solar:a3.1_slot_bracket_left,

aspecto−fr:has_component Aspect−FR−Solar:a3.1_slot_bracket_right,

aspecto−fr:s_connected_to Aspect−FR−Solar:a2.1_mounting_base_left,

aspecto−fr:s_connected_to Aspect−FR−Solar:a2.1_mounting_base_right,

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:a3.1_screw_bracket_set,

Aspect−FR−Solar:has_manufacturing_cost 5.43

Individual: Aspect−FR−Solar:a4.1_ballast_tray_pad_left

Annotations:

rdfs:comment "Ballast tray pad"@en

Types:

Aspect−FR−Solar:Ballast_tray_pad

Facts:

aspecto−fr:s_directly_connected_to Aspect−FR−Solar:a9.3_deck_bay_1−2,
aspecto−fr:s_directly_connected_to Aspect−FR−Solar:a9.4_roof_membrane,
Aspect−FR−Solar:has_manufacturing_cost .11

Individual: Aspect−FR−Solar:a4.1_ballast_tray_pad_right

Annotations:

rdfs:label "Ballast tray pad"@en

Types:

Aspect−FR−Solar:Ballast_tray_pad

Facts:

aspecto−fr:s_connected_to Aspect−FR−Solar:a9.3_deck_bay_2−3,
aspecto−fr:s_directly_connected_to Aspect−FR−Solar:a9.4_roof_membrane,
Aspect−FR−Solar:has_manufacturing_cost .11

Individual: Aspect−FR−Solar:a4.2_ballast_left

Annotations:

rdfs:comment "Ballast"@en

Types:

Aspect−FR−Solar:Ballast

Facts:

aspecto−fr:s_directly_connected_to Aspect−FR−Solar:a4_ballast_tray_left,
Aspect−FR−Solar:has_manufacturing_cost 0.2

Individual: Aspect−FR−Solar:a4.2_ballast_right

Annotations:

rdfs:comment "Ballast"@en

Types:

Aspect−FR−Solar:Ballast

Facts:

aspecto−fr:s_directly_connected_to Aspect−FR−Solar:a4_ballast_tray_right,
Aspect−FR−Solar:has_manufacturing_cost 0.2

Individual: Aspect−FR−Solar:a4_ballast_tray_left

Annotations:

rdfs:comment "Ballast tray"@en

Types:

Aspect−FR−Solar:Ballast_tray

Facts:

aspecto−fr:s_directly_connected_to Aspect−FR−Solar:a4.1_ballast_tray_pad_left,

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:a4.1_ballast_tray_pad_left,

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:a4.2_ballast_left,

Aspect−FR−Solar:has_manufacturing_cost 1.045,

Aspect−FR−Solar:has_weight 20

Individual: Aspect−FR−Solar:a4_ballast_tray_right

Annotations:

rdfs:label "Ballast tray"@en

Types:

Aspect−FR−Solar:Ballast_tray

Facts:

aspecto−fr:s_directly_connected_to Aspect−FR−Solar:a4.1_ballast_tray_pad_right,

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:a4.1_ballast_tray_pad_right,

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:a4.2_ballast_right,

Aspect−FR−Solar:has_manufacturing_cost 1.045,

Aspect−FR−Solar:has_weight 40

Individual: Aspect−FR−Solar:a5_inverter

Types:

Aspect−FR−Solar:Electrical_component

Individual: Aspect−FR−Solar:a5_junction_box

Types:

Aspect−FR−Solar:Electrical_component

Individual: Aspect−FR−Solar:a5_wires

Types:

Aspect−FR−Solar:Electrical_component

Facts:

DOLCE−Lite−FR:participant−in Aspect−FR−Solar:e3.1_wiring,

DOLCE−Lite−FR:participant−in Aspect−FR−Solar:e3_electrical_installation

Individual: Aspect−FR−Solar:a5_zip_ties

Types:

Aspect−FR−Solar:Fastener

Facts:

DOLCE−Lite−FR:participant−in Aspect−FR−Solar:e3.1_wiring

Individual: Aspect−FR−Solar:a6_pv_module

Annotations:

rdfs:comment "PV module"@en

Types:

Aspect−FR−Solar:PV_module

Facts:

DOLCE−Lite−FR:participant−in Aspect−FR−Solar:e2.4_PV_module_attachment,

DOLCE−Lite−FR:participant−in Aspect−FR−Solar:e4.1_uplift,

DOLCE−Lite−FR:participant−in Aspect−FR−Solar:e_bracing

Individual: Aspect−FR−Solar:a7.1_tape_measure

Facts:

aspecto−fr:nominal_participant_in Aspect−FR−Solar:e5.2_taking_measures

Individual: Aspect−FR−Solar:a7.2_chalk_line

Facts:

aspecto−fr:nominal_participant_in Aspect−FR−Solar:e5.3_laying_chalk_lines

Individual: Aspect−FR−Solar:a7_squaring_equipment

Facts:

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:a7.1_tape_measure,

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:a7.2_chalk_line

Individual: Aspect−FR−Solar:a8_battery

Facts:

aspecto−fr:_b_has_capability Aspect−FR−Solar:b_portable_power_supply

Individual: Aspect−FR−Solar:a8_charger

Facts:

aspecto−fr:_b_has_capability Aspect−FR−Solar:b_charging_capability

Individual: Aspect−FR−Solar:a8_extension_cord

Facts:

aspecto−fr:_b_has_capability Aspect−FR−Solar:b_conductivity

Individual: Aspect−FR−Solar:a8_installation_tools

Facts:

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:a8_screw_driver_1,

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:a8_screw_driver_2

Individual: Aspect−FR−Solar:a8_power_outlet

Facts:

aspecto−fr:_b_has_capability Aspect−FR−Solar:b_supply_AC_power

Individual: Aspect−FR−Solar:a8_screw_driver_1

Types:

Aspect−FR−Solar:Screw_driver

Facts:

aspecto−fr:_b_has_capability Aspect−FR−Solar:b_manual_screwing,

Aspect−FR−Solar:has_installation_speed "slow",

Aspect−FR−Solar:has_torque_control false

Individual: Aspect−FR−Solar:a8_screw_driver_2

Types:

Aspect−FR−Solar:Screw_driver

Facts:

aspecto−fr:_b_has_capability Aspect−FR−Solar:b_battery_powered,

DOLCE−Lite−FR:participant−in Aspect−FR−Solar:e3_electrical_installation,

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:a8_battery

Individual: Aspect−FR−Solar:a9.1_roof_assembly

Facts:

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:a9.2_roof_structure,

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:a9.3_roof_deck,

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:a9.4_roof_membrane

Individual: Aspect−FR−Solar:a9.2_joist_1

Types:

Aspect−FR−Solar:Roof_joist

Individual: Aspect−FR−Solar:a9.2_joist_2

Types:

Aspect−FR−Solar:Roof_joist

Individual: Aspect−FR−Solar:a9.2_joist_3

Types:

Aspect−FR−Solar:Roof_joist

Individual: Aspect−FR−Solar:a9.2_joist_4

Types:

Aspect−FR−Solar:Roof_joist

Individual: Aspect−FR−Solar:a9.2_roof_structure

Annotations:

rdfs:comment "Roof structure"@en

Types:

Aspect−FR−Solar:Roof_structure

Facts:

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:a9.2_joist_1,
DOLCE−Lite−FR:proper−part Aspect−FR−Solar:a9.2_joist_2,
DOLCE−Lite−FR:proper−part Aspect−FR−Solar:a9.2_joist_3,
DOLCE−Lite−FR:proper−part Aspect−FR−Solar:a9.2_joist_4

Individual: Aspect−FR−Solar:a9.3_deck_bay_1−2

Types:

Aspect−FR−Solar:Roof_deck_bay

Facts:

aspecto−fr:s_directly_connected_to Aspect−FR−Solar:a9.2_joist_1,

aspecto−fr:s_directly_connected_to Aspect−FR−Solar:a9.2_joist_2

Individual: Aspect−FR−Solar:a9.3_deck_bay_2−3

Types:

Aspect−FR−Solar:Roof_deck_bay

Facts:

aspecto−fr:s_directly_connected_to Aspect−FR−Solar:a9.2_joist_2,

aspecto−fr:s_directly_connected_to Aspect−FR−Solar:a9.2_joist_3

Individual: Aspect−FR−Solar:a9.3_deck_bay_3−4

Types:

Aspect−FR−Solar:Roof_deck_bay

Facts:

aspecto−fr:s_directly_connected_to Aspect−FR−Solar:a9.2_joist_3,

aspecto−fr:s_directly_connected_to Aspect−FR−Solar:a9.2_joist_4

Individual: Aspect−FR−Solar:a9.3_roof_deck

Annotations:

rdfs:comment "Roof substrate"@en

Types:

Aspect−FR−Solar:Roof_deck,

Aspect−FR−Solar:Roof_substrate

Facts:

aspecto−fr:has_component Aspect−FR−Solar:a9.3_deck_bay_1−2,

aspecto−fr:has_component Aspect−FR−Solar:a9.3_deck_bay_2−3,

aspecto−fr:has_component Aspect−FR−Solar:a9.3_deck_bay_3−4,

aspecto−fr:s_directly_connected_to Aspect−FR−Solar:a9.2_roof_structure

Individual: Aspect−FR−Solar:a9.4_roof_membrane

Annotations:

rdfs:comment "Roof membrane"@en

Types:

Aspect−FR−Solar:Roof_membrane

Facts:

aspecto−fr:s_directly_connected_to Aspect−FR−Solar:a9.3_roof_deck,

DOLCE−Lite−FR:participant−in Aspect−FR−Solar:e2.2b_pre−squaring,

DOLCE−Lite−FR:participant−in Aspect−FR−Solar:e_sliding

Individual: Aspect−FR−Solar:a9.5_parapet

Facts:

DOLCE−Lite−FR:participant−in Aspect−FR−Solar:e_loading_wind

Individual: Aspect−FR−Solar:a9_building

Annotations:

rdfs : label "Building"@en

Types:

Aspect−FR−Solar:Building

Facts:

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:a1_pv_racking,

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:a9.1_roof_assembly,

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:a9.5_parapet

Individual: Aspect−FR−Solar:a_nothing

Types:

Aspect−FR−Solar:Aspect_system_squaring_function

Individual: Aspect−FR−Solar:b_anti_corrosion_capability

Types:

aspecto−fr:Behavior

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Solar:e_corrosion,

Aspect−FR−Solar:has_anti−corrosion_property "galvanization",

Aspect−FR−Solar:has_galvanization "R60"

Individual: Aspect−FR−Solar:b_ballasted

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Solar:e2.3b_ballasted,

Aspect−FR−Solar:has_weight_per_area 2.75

Individual: Aspect−FR−Solar:b_battery_powered

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Solar:e_power_screwing,
Aspect−FR−Solar:has_installation_speed "fast",
Aspect−FR−Solar:has_torque 30,
Aspect−FR−Solar:has_torque_control true,
Aspect−FR−Solar:has_voltage_required 20

Individual: Aspect−FR−Solar:b_bonding_by_torque

Types:

Aspect−FR−Solar:Bonding_capability

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Solar:e_electrical_continuity,
Aspect−FR−Solar: has_electrical_resistitivity 20

Individual: Aspect−FR−Solar:b_bracing_capability

Types:

Aspect−FR−Solar:Bracing_capability

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Solar:e_bracing,
Aspect−FR−Solar:has_compression_capability 50,
Aspect−FR−Solar:has_tensile_capability 65

Individual: Aspect−FR−Solar:b_charging_capability

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Solar:e_charge,

Aspect−FR−Solar:has_charging_time 60,

Aspect−FR−Solar:has_voltage_required 220

Individual: Aspect−FR−Solar:b_conductivity

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Solar:e_charge,

Annotations: rdfs:label "gauge"^^xsd:string

Aspect−FR−Solar:has_conductor_gauge "12/3",

Aspect−FR−Solar:has_extension 30,

Annotations: rdfs:label "Amps"^^xsd:string

Aspect−FR−Solar:has_max_amps 15

Individual: Aspect−FR−Solar:b_electrical_tested

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Solar:e3.4_electrical_system_tested

Individual: Aspect−FR−Solar:b_flat_packing

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Solar:e1.1_transporting

Individual: Aspect−FR−Solar:b_flat_unpacking

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Solar:e1_hardware_unpacking,
Aspect−FR−Solar:has_installation_advantage "no waste"

Individual: Aspect−FR−Solar:b_grounding

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Solar:e_electrical_grounding

Individual: Aspect−FR−Solar:b_hardware_assembling

Individual: Aspect−FR−Solar:b_hardware_installed

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Solar:e2.5_racking_installed

Individual: Aspect−FR−Solar:b_integrated_squaring

Types:

Aspect−FR−Solar:Squaring_capability

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Solar:e2.2b_pre−squaring,

Aspect−FR−Solar:has_ergonomic_advantage "yes",

Annotations: rdfs:label "Seconds"

Aspect−FR−Solar:has_installation_time 45

Individual: Aspect−FR−Solar:b_integrated_wire_bundling

Types:

Aspect−FR−Solar:Wiring_capability

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Solar:e3.1_wires_bundled,

Aspect−FR−Solar:has_ergonomic_advantage "yes"

Individual: Aspect−FR−Solar:b_integrated_wire_management

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Solar:e3.2_wire_management

Individual: Aspect−FR−Solar:b_keep_air_pressure_balance

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Solar:e4.3_air_pressure_balance

Individual: Aspect−FR−Solar:b_loading_seismic

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Solar:e_loading_seismic

Individual: Aspect−FR−Solar:b_loading_wind

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Solar:e_loading_wind,
Aspect−FR−Solar:has_wind_speed_limit 90

Individual: Aspect−FR−Solar:b_maintain_form_capability

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Solar:e7.1_maintain_form,
Aspect−FR−Solar:has_bending_moment 167,
Aspect−FR−Solar:has_compression_capability 55,
Aspect−FR−Solar:has_tensile_capability 77.5

Individual: Aspect−FR−Solar:b_maintain_position_capability

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Solar:e7.1_maintain_position,
Aspect−FR−Solar:has_weight_per_area 2.7,
Aspect−FR−Solar:has_wind_speed_limit 65

Individual: Aspect−FR−Solar:b_maintain_s_integrity_capability

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Solar:e7_final

Individual: Aspect−FR−Solar:b_manual_screwing

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Solar:e_screwing_3,
Aspect−FR−Solar:has_installation_speed "slow",
Aspect−FR−Solar:has_torque_control false

Individual: Aspect−FR−Solar:b_measuring

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Solar:e5_pre−
squaring_with_measure

Individual: Aspect−FR−Solar:b_moment_resistence_capability

Types:

Aspect−FR−Solar:Moment_resistant_capability

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Solar:e_moment_transfer,
Aspect−FR−Solar:has_bending_moment 725

Individual: Aspect−FR−Solar:b_moment_transfer_capability

Types:

Aspect−FR−Solar:Moment_resistant_capability

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Solar:e_moment_transfering

Individual: Aspect−FR−Solar:b_normal_squaring

Types:

Aspect−FR−Solar:Squaring_capability

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Solar:e2.2b_pre−squaring,

Aspect−FR−Solar:has_installation_time 120

Individual: Aspect−FR−Solar:b_portable_power_supply

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Solar:e_charge,

Aspect−FR−Solar:has_supply_cycles 500

Individual: Aspect−FR−Solar:b_pv_module_attachment

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Solar:e2.4_PV_module_attachment

Individual: Aspect−FR−Solar:b_quick_fastening

Types:

Aspect−FR−Solar:Fastening_capability

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Solar:e_slot_fastened,

aspecto−fr:behavioral_constraint_on Aspect−FR−Solar:pb_installation_speed,

Aspect−FR−Solar:has_installation_time 30

Individual: Aspect−FR−Solar:b_reduces_drag_by_friction

Types:

Aspect−FR−Solar:Drag_resistance_capability

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Solar:e_sliding,

Aspect−FR−Solar:has_friction_coefficient 0.16

Individual: Aspect−FR−Solar:b_reduces_drag_by_weight

Types:

Aspect−FR−Solar:Drag_resistance_capability

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Solar:e4.2_drag,

Aspect−FR−Solar:has_weight_per_area 2.7

Individual: Aspect−FR−Solar:b_reduces_uplift_by_deflection

Types:

Aspect−FR−Solar:Uplift_resistance_capability

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Solar:e4.1_uplift,

Aspect−FR−Solar:has_aerodynamic_coefficient 0.09

Individual: Aspect−FR−Solar:b_reduces_uplift_by_weight

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Solar:e4.1_uplift,

Aspect−FR−Solar:has_weight_per_area 2.75

Individual: Aspect−FR−Solar:b_screwed_slotted

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Solar:e_screwed_fastened,

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Solar:e_slot_fastened

Individual: Aspect−FR−Solar:b_screwing

Types:

Aspect−FR−Solar:Fastening_capability

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Solar:e_screwed_fastened

Individual: Aspect−FR−Solar:b_screwing_1

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Solar:e_screwing_1

Individual: Aspect−FR−Solar:b_slotting_1

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Solar:e_slotting_1

Individual: Aspect−FR−Solar:b_slotting_2

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Solar:b_slotting_2

Individual: Aspect−FR−Solar:b_some_squaring

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Solar:e2.2b_pre−squaring

Individual: Aspect−FR−Solar:b_squared

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Solar:e2.3a_squared

Individual: Aspect−FR−Solar:b_staged

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Solar:e2.1_hardware_staged

Individual: Aspect−FR−Solar:b_supply_AC_power

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Solar:e_charge,

Aspect−FR−Solar:has_power_supply 110

Individual: Aspect−FR−Solar:b_supply_DC_power

Individual: Aspect−FR−Solar:b_thermal_expansion_coefficient

Types:

aspecto−fr:Behavior

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Solar:e_thermal_expansion,

Aspect−FR−Solar:has_thermal_expansion_coefficient 7.2

Individual: Aspect−FR−Solar:b_tighted_by_torque

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Solar:e_screwed_fastened

Individual: Aspect−FR−Solar:b_wiring

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Solar:e3.1_wiring

Individual: Aspect−FR−Solar:e1.1_transporting

Annotations:

rdfs : label "packaging / flat −packing"@en

Types:

DOLCE−Lite−FR:state

Facts:

aspecto−fr:_c_causes Aspect−FR−Solar:e1_hardware_unpacking

Individual: Aspect−FR−Solar:e1_hardware_unpacking

Annotations:

rdfs : label "unpacking"@en

Types:

Aspect−FR−Solar:Unpacked

Facts:

aspecto−fr:_c_causes Aspect−FR−Solar:e2.1_hardware_staged

Individual: Aspect−FR−Solar:e1_staging

Annotations:

rdfs : label "storage / transportation"

Facts:

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:e1.1_transporting,

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:e1_hardware_unpacking

Individual: Aspect−FR−Solar:e2.1_hardware_staged

Annotations:

rdfs:comment "staging"

Facts:

aspecto−fr:_d_causes Aspect−FR−Solar:e2.2a_pre−attachment

Individual: Aspect−FR−Solar:e2.2a_pre−attachment

Annotations:

rdfs:comment "wind deflector attachment"

Facts:

aspecto−fr:_d_causes Aspect−FR−Solar:e2.2b_pre−squaring

Individual: Aspect−FR−Solar:e2.2b_pre−squaring

Annotations:

rdfs:comment "pre−squaring"

Individual: Aspect−FR−Solar:e2.3a_squared

Types:

Aspect−FR−Solar:Squared

Facts:

aspecto−fr:_d_causes Aspect−FR−Solar:e2.3b_ballasted

Individual: Aspect−FR−Solar:e2.3b_ballasted

Facts:

aspecto−fr:_c_causes Aspect−FR−Solar:e2.4_PV_module_attachment

Individual: Aspect−FR−Solar:e2.4_PV_module_attachment

Annotations:

rdfs:label "PV module attachment"@en

Types:

Aspect−FR−Solar:Attachment

Facts:

aspecto−fr:_d_causes Aspect−FR−Solar:e2.5_racking_installed

Individual: Aspect−FR−Solar:e2.5_racking_installed

Annotations:

rdfs:comment "hardware installation completed"

Types:

DOLCE−Lite−FR:state

Facts:

aspecto−fr:_c_causes Aspect−FR−Solar:e3.1_wiring

Individual: Aspect−FR−Solar:e2_hardware_installation

Annotations:

rdfs : label "hardware installation"@en

Facts:

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:e2.1_hardware_staged,

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:e2.4_PV_module_attachment,

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:e2.5_racking_installed

Individual: Aspect−FR−Solar:e3.1_wires_bundled

Types:

Aspect−FR−Solar:Wired

Individual: Aspect−FR−Solar:e3.1_wiring

Annotations:

rdfs : label "wiring"@en

Facts:

aspecto−fr:_d_causes Aspect−FR−Solar:e3.1_wires_bundled

Individual: Aspect−FR−Solar:e3.2_wire_management

Annotations:

rdfs : label "wire management"@en

Types:

Aspect−FR−Solar:Wired

Individual: Aspect−FR−Solar:e3.3_eletrical_testing

Annotations:

rdfs:comment "electrical connection testing"

Facts:

aspecto−fr:_d_causes Aspect−FR−Solar:e3.4_electrical_system_tested

Individual: Aspect−FR−Solar:e3.4_electrical_system_tested

Annotations:

rdfs:label " electrical installation completed"@en

Types:

DOLCE−Lite−FR:state

Individual: Aspect−FR−Solar:e3_electrical_installation

Annotations:

rdfs:comment "electrical installation "

Facts:

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:e3.1_wiring,

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:e3.3_eletrical_testing,

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:e3.4_electrical_system_tested,

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:e_electrical_grounding

Individual: Aspect−FR−Solar:e4.1_uplift

Annotations:

rdfs:comment "uplift"

Types:

Aspect−FR−Solar:Uplifting

Individual: Aspect−FR−Solar:e4.2_drag

Types:

Aspect−FR−Solar:Dragging

Facts:

aspecto−fr:_c_causes Aspect−FR−Solar:e_sliding

Individual: Aspect−FR−Solar:e4.3_air_pressure_balance

Facts:

aspecto−fr:_c_causes Aspect−FR−Solar:e4.1_uplift

Individual: Aspect−FR−Solar:e5.1_walking

Facts:

aspecto−fr:_c_causes Aspect−FR−Solar:e5.2_taking_measures

Individual: Aspect−FR−Solar:e5.2_taking_measures

Facts:

aspecto−fr:_d_causes Aspect−FR−Solar:e5.3_laying_chalk_lines

Individual: Aspect−FR−Solar:e5.3_laying_chalk_lines

Facts:

aspecto−fr:_d_causes Aspect−FR−Solar:e5_pre−squaring_with_measure

Individual: Aspect−FR−Solar:e5_pre−squaring_with_measure

Facts:

aspecto−fr:_d_causes Aspect−FR−Solar:e2.2b_pre−squaring,
DOLCE−Lite−FR:proper−part Aspect−FR−Solar:e5.1_walking,
DOLCE−Lite−FR:proper−part Aspect−FR−Solar:e5.2_taking_measures,
DOLCE−Lite−FR:proper−part Aspect−FR−Solar:e5.3_laying_chalk_lines

Individual: Aspect−FR−Solar:e7.1_maintain_form

Types:

Aspect−FR−Solar:Maintain_form

Facts:

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:e_bracing,
DOLCE−Lite−FR:proper−part Aspect−FR−Solar:e_buckling

DifferentFrom:

Aspect−FR−Solar:e7.1_maintain_position

Individual: Aspect−FR−Solar:e7.1_maintain_position

Annotations:

rdfs:comment "maintain position / structural stability"

Types:

Aspect−FR−Solar:Maintain_position

Facts:

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:e4.1_uplift,

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:e4.2_drag

DifferentFrom:

Aspect−FR−Solar:e7.1_maintain_form

Individual: Aspect−FR−Solar:e7_final

Individual: Aspect−FR−Solar:e7_maintain_form_and_position

Facts:

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:e7.1_maintain_form,

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:e7.1_maintain_position

Individual: Aspect−FR−Solar:e_ballasting

Individual: Aspect−FR−Solar:e_bracing

Types:

Aspect−FR−Solar:Bracing

Facts:

aspecto−fr:_c_causes Aspect−FR−Solar:e_buckling,

aspecto−fr:_d_causes Aspect−FR−Solar:e7.1_maintain_form

Individual: Aspect−FR−Solar:e_buckling

Types:

Aspect−FR−Solar:Buckling

Facts:

aspecto−fr:_c_causes Aspect−FR−Solar:e7.1_maintain_form

Individual: Aspect−FR−Solar:e_charge

Facts:

aspecto−fr:_c_causes Aspect−FR−Solar:e_power_screwing

Individual: Aspect−FR−Solar:e_compression

Facts:

aspecto−fr:_c_causes Aspect−FR−Solar:e_bracing,

aspecto−fr:_c_causes Aspect−FR−Solar:e_sliding

Individual: Aspect−FR−Solar:e_corrosion

Facts:

aspecto−fr:_c_causes Aspect−FR−Solar:e7.1_maintain_form

Individual: Aspect−FR−Solar:e_electrical_conductivity

Facts:

aspecto−fr:_c_causes Aspect−FR−Solar:e_electrical_grounding

Individual: Aspect−FR−Solar:e_electrical_continuity

Annotations:

rdfs:comment "This epd is not asserted to cause anything, because causality is inferred from equivalent class Grounding_E"

Types:

Aspect−FR−Solar:Bonding

Individual: Aspect−FR−Solar:e_electrical_grounding

Types:

Aspect−FR−Solar:Grounding

Facts:

aspecto−fr:_c_causes Aspect−FR−Solar:e3.3_eletrical_testing

Individual: Aspect−FR−Solar:e_global

Annotations:

rdfs:comment "installation"

Facts:

aspecto−fr:_EF_inPD Aspect−FR−Solar:e1.1_transporting,

aspecto−fr:_EF_outPD Aspect−FR−Solar:e3.4_electrical_system_tested

Individual: Aspect−FR−Solar:e_gravity

Facts:

aspecto−fr:_c_causes Aspect−FR−Solar:e4.1_uplift

Individual: Aspect−FR−Solar:e_loading_seismic

Facts:

aspecto−fr:_c_causes Aspect−FR−Solar:e_compression,

aspecto−fr:_c_causes Aspect−FR−Solar:e_sliding

Individual: Aspect−FR−Solar:e_loading_wind

Facts:

aspecto−fr:_c_causes Aspect−FR−Solar:e_compression,

aspecto−fr:_d_causes Aspect−FR−Solar:e4.1_uplift

Individual: Aspect−FR−Solar:e_moment_transfer

Individual: Aspect−FR−Solar:e_moment_transfering

Individual: Aspect−FR−Solar:e_piercing

Facts:

aspecto−fr:_d_causes Aspect−FR−Solar:e_electrical_continuity

Individual: Aspect−FR−Solar:e_power_screwing

Facts:

aspecto−fr:_d_causes Aspect−FR−Solar:e_screwing_3

Individual: Aspect−FR−Solar:e_screwed_fastened

Types:

Aspect−FR−Solar:Fastened

Individual: Aspect−FR−Solar:e_screwing_1

Facts:

aspecto−fr:_d_causes Aspect−FR−Solar:e_screwing_2

Individual: Aspect−FR−Solar:e_screwing_2

Facts:

aspecto−fr:_d_causes Aspect−FR−Solar:e_screwing_3

Individual: Aspect−FR−Solar:e_screwing_3

Facts:

aspecto−fr:_d_causes Aspect−FR−Solar:e_screwed_fastened

Individual: Aspect−FR−Solar:e_sliding

Types:

<http://www.ou.edu/coa/ontologies/2018/6/Aspect−FR−KB#Sliding>

Individual: Aspect−FR−Solar:e_slot_fastened

Types:

Aspect−FR−Solar:Fastened

Individual: Aspect−FR−Solar:e_slotting_1

Facts:

aspecto−fr:_d_causes Aspect−FR−Solar:e_slot_fastened

Individual: Aspect−FR−Solar:e_slotting_2

Facts:

aspecto−fr:_d_causes Aspect−FR−Solar:e_slot_fastened

Individual: Aspect−FR−Solar:e_thermal_expansion

Facts:

aspecto−fr:_c_causes Aspect−FR−Solar:e_buckling

Individual: Aspect−FR−Solar:life_a1

Facts:

DOLCE−Lite−FR:proper−part Aspect−FR−Solar:e7.1_maintain_position

Individual: Aspect−FR−Solar:life_a3

Individual: Aspect−FR−Solar:pb_installation_speed

Types:

<http://www.ou.edu/coa/ontologies/2018/6/Aspect−FR−KB#Increase>

Individual: Aspect−FR−Solar:st1

Types:

aspecto−fr:rational−physical−object

DifferentIndividuals :

Aspect−FR−Solar:a3.2a_screw_1,Aspect−FR−Solar:a3.2a_screw_2,Aspect−FR−Solar:a3.2b_screw_2

APPENDIX F

CASE STUDY 4: CO-WORKING SPACE

Prefix: Aspect−FR−Coworking: <http://www.ou.edu/coa/ontologies/2018/4/Aspect−FR−Coworking#>

Prefix: aspecto−fr: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR#>

Prefix: DOLCE−Lite−FR: <http://www.ou.edu/coa/ontologies/DOLCE−Lite−FR.owl#>

Prefix: owl: <http://www.w3.org/2002/07/owl#>

Prefix: rdf: <http://www.w3.org/1999/02/22−rdf−syntax−ns#>

Prefix: rdfs: <http://www.w3.org/2000/01/rdf−schema#>

Prefix: swrla: <http://swrl.stanford.edu/ontologies/3.3/swrla.owl#>

Prefix: xml: <http://www.w3.org/XML/1998/namespace>

Prefix: xsd: <http://www.w3.org/2001/XMLSchema#>

Ontology: <http://www.ou.edu/coa/ontologies/2018/4/Aspect−FR−Coworking>

Import: <http://www.ou.edu/coa/ontologies/2018/6/Aspecto−FR>

Import: <http://www.ou.edu/coa/ontologies/2018/6/Aspect−FR−KB>

AnnotationProperty: rdfs:comment

AnnotationProperty: rdfs:label

Datatype: rdf:PlainLiteral

Datatype: xsd:decimal

ObjectProperty: aspecto−fr:_DF_artifact

ObjectProperty: aspecto−fr:_DF_outPD

ObjectProperty: aspecto−fr:_EF_in_environment

ObjectProperty: aspecto−fr:_EF_in_mode_of_deployment

ObjectProperty: aspecto−fr:_EF_nominal_artifact

ObjectProperty: aspecto−fr:_EF_outPD

ObjectProperty: aspecto−fr:_b_has_capability

ObjectProperty: aspecto−fr:_c_caused_by

ObjectProperty: aspecto−fr:_c_causes

ObjectProperty: aspecto−fr:_d_causes

ObjectProperty: aspecto−fr:b0

ObjectProperty: aspecto−fr:b1

ObjectProperty: aspecto−fr:behavior_constraint_on_perdurant

ObjectProperty: aspecto−fr:causal_participant_in

ObjectProperty: aspecto−fr:functional_role_in

ObjectProperty: aspecto−fr:has_daemon

ObjectProperty: aspecto−fr:nominal_participant_in

ObjectProperty: aspecto−fr:s_directly_connected_to

ObjectProperty: DOLCE−Lite−FR:part

ObjectProperty: DOLCE−Lite−FR:participant

ObjectProperty: DOLCE−Lite−FR:participant−in

ObjectProperty: DOLCE−Lite−FR:proper−part

ObjectProperty: DOLCE−Lite−FR:proper−part−of

ObjectProperty: DOLCE−Lite−FR:weak−connection

DataProperty: Aspect−FR−Coworking:relative_humidity

Annotations:

rdfs:label "RH"

DataProperty: Aspect−FR−Coworking:air_temperature

Annotations:

rdfs:label "Celsius"

DataProperty: Aspect−FR−Coworking:dew_point_temperature

Class: Aspect−FR−Coworking:Animal

Annotations:

rdfs:label "Animal"

SubClassOf:

Aspect−FR−Coworking:Living,

aspecto−fr:_b_has_capability value Aspect−FR−Coworking:b_transpiration_by_activity

Class: Aspect−FR−Coworking:Animal_metabolism

Annotations:

rdfs:label "Animal_metabolism"

SubClassOf:

Aspect−FR−Coworking:Metabolism

Class: Aspect−FR−Coworking:Aspect_system_condensation_failure

Annotations:

rdfs:comment "(part some (participant−in some (_c_causes some (inverse (_DF_outPD) value fe_failure_mode_condensation))))
and (proper−part−of value Env1)"

EquivalentTo:

(DOLCE−Lite−FR:part some (DOLCE−Lite−FR:participant−in some (aspecto−fr:_c_causes some (inverse (aspecto−fr:_DF_outPD) value Aspect−FR−Coworking:fe_failure_mode_condensation))))
and (DOLCE−Lite−FR:proper−part−of value <http://www.ou.edu/coa/ontologies/2018/6/Aspect−FR−KB#Env1>)

Class: Aspect−FR−Coworking:Aspect_system_productivity

EquivalentTo:

(DOLCE−Lite−FR:part some (DOLCE−Lite−FR:participant−in some (aspecto−fr:_c_causes some (aspecto−fr:functional_role_in some (inverse (aspecto−fr:_EF_outPD) value Aspect−FR−Coworking:fe_productivity)))))
and (DOLCE−Lite−FR:proper−part−of value <http://www.ou.edu/coa/ontologies/2018/6/Aspect−FR−KB#Env1>)

Class: Aspect−FR−Coworking:Aspect_system_productivity_full

EquivalentTo:

Aspect−FR−Coworking:Aspect_system_productivity or ((DOLCE−Lite−FR:part some (DOLCE−Lite−FR:participant−in some (aspecto−fr:functional_role_in some (inverse (aspecto−fr:_EF_outPD) value Aspect−FR−Coworking:fe_productivity))))
and (DOLCE−Lite−FR:proper−part−of value <http://www.ou.edu/coa/ontologies/2018/6/Aspect−FR−KB#Env1>))

Class: Aspect−FR−Coworking:Boiler

Annotations:

rdfs : label "Boiler"

SubClassOf:

aspecto−fr: Technical_artifact ,

DOLCE−Lite−FR:participant−in some Aspect−FR−Coworking:Boiling_water

Class: Aspect−FR−Coworking:Boiling_water

Annotations:

rdfs : label "Boiling_water"

SubClassOf:

DOLCE−Lite−FR:process,

aspecto−fr:_d_causes value Aspect−FR−Coworking:e_evaporation

Class: Aspect−FR−Coworking:Condensable_surface

Annotations:

rdfs : label "Condensable_surface"

EquivalentTo:

(aspecto−fr: s_directly_connected_to some

((DOLCE−Lite−FR:participant−in value Aspect−FR−Coworking:e_moisture_transfer)

and (Aspect−FR−Coworking:dew_point_temperature some xsd:decimal[> 16])))

and (aspecto−fr: s_directly_connected_to some (Aspect−FR−Coworking:air_temperature

some xsd:decimal[< 20]))

SubClassOf:

DOLCE−Lite−FR:participant−in value e_moisture_retention

Class: Aspect−FR−Coworking:Condensation

Annotations:

rdfs : label "Condensation"

EquivalentTo:

DOLCE−Lite−FR:participant some Aspect−FR−Coworking:Condensable_surface

SubClassOf:

aspecto−fr:CauseMD,

aspecto−fr:_c_caused_by value Aspect−FR−Coworking:e_moisture_transfer

Class: Aspect−FR−Coworking:Daemon

SubClassOf:

aspecto−fr:agentive−physical−object

Class: Aspect−FR−Coworking:Daemon_condensation

Annotations:

rdfs:comment "Daemon_condensation"

EquivalentTo:

Aspect−FR−Coworking:Humidifiers

and (aspecto−fr:s_directly_connected_to value Aspect−FR−Coworking:a1.4_indoor_air)

SubClassOf:

Aspect−FR−Coworking:Daemon,

inverse (aspecto−fr:_DF_artifact) value Aspect−FR−Coworking:

fd_humidifier_daemon_function

Class: Aspect−FR−Coworking:Humid_air

EquivalentTo:

aspecto−fr:Air

and (DOLCE−Lite−FR:weak−connection some Aspect−FR−Coworking:Humidifiers)

and (DOLCE−Lite−FR:proper−part−of value Aspect−FR−Coworking:a1_office_building

)

SubClassOf:

DOLCE−Lite−FR:participant−in value Aspect−FR−Coworking:e_moisture_transfer,

Aspect−FR−Coworking:dew_point_temperature value 18

Class: Aspect−FR−Coworking:Humidifiers

EquivalentTo:

aspecto−fr:causal_participant_in value Aspect−FR−Coworking:e_moisture_absortion

Class: Aspect−FR−Coworking:Living

SubClassOf:

aspecto−fr:agentive−physical−object,

DOLCE−Lite−FR:participant−in some Aspect−FR−Coworking:Metabolism

Class: Aspect−FR−Coworking:Metabolism

SubClassOf:

DOLCE−Lite−FR:process,

aspecto−fr:_d_causes value Aspect−FR−Coworking:e_transpiration

Class: Aspect−FR−Coworking:Moisture_transfer

Annotations:

rdfs : label "Moisture_transfer"

EquivalentTo:

DOLCE−Lite−FR:participant some (DOLCE−Lite−FR:weak−connection some Aspect−FR−Coworking:Humidifiers)

SubClassOf:

aspecto−fr:CauseMD,

aspecto−fr:_c_caused_by value Aspect−FR−Coworking:e_moisture_absortion

Class: Aspect−FR−Coworking:Realizable_failure_mode

EquivalentTo:

aspecto−fr:_EF_nominal_artifact some Aspect−FR−Coworking:Daemon_condensation

SubClassOf:

aspecto−fr:Realizable_E_Function

Class: Aspect−FR−Coworking:Satisfiable_daemon_function

EquivalentTo:

aspecto−fr:_DF_artifact some Aspect−FR−Coworking:Daemon_condensation

SubClassOf:

aspecto−fr: Satisfiable_D_function

Class: Aspect−FR−Coworking:Vegetal

Annotations:

rdfs : label "Vegetal"

SubClassOf:

Aspect−FR−Coworking:Living,

aspecto−fr: _b_has_capability value Aspect−FR−Coworking:b_by_photosynthesis

Class: Aspect−FR−Coworking:Vegetal_metabolism

Annotations:

rdfs : label "Vegetal_metabolism"

SubClassOf:

Aspect−FR−Coworking:Metabolism

Class: aspecto−fr:Air

Class: aspecto−fr:CauseMD

Class: aspecto−fr:OutPD

Class: aspecto−fr:Realizable_E_Function

Class: aspecto−fr: Satisfiable_D_function

Class: aspecto−fr: Technical_artifact

Class: aspecto−fr:Technical_system

Class: aspecto−fr:_E_function

Class: aspecto−fr:agentive−physical−object

Class: DOLCE−Lite−FR:process

Individual: <http://www.ou.edu/coa/ontologies/2018/6/Aspect−FR−KB#Env1>

Facts:

DOLCE−Lite−FR:proper−part Aspect−FR−Coworking:a0_outdoor_space,

DOLCE−Lite−FR:proper−part Aspect−FR−Coworking:a1_office_building,

DOLCE−Lite−FR:proper−part Aspect−FR−Coworking:a_daemon

Individual: Aspect−FR−Coworking:a0.1_outdoor_air

Types:

aspecto−fr:Air

Facts:

Aspect−FR−Coworking:air_temperature 15

DifferentFrom:

Aspect−FR−Coworking:a1.4_indoor_air

Individual: Aspect−FR−Coworking:a0_outdoor_space

Facts:

DOLCE−Lite−FR:proper−part Aspect−FR−Coworking:a0.1_outdoor_air

Individual: Aspect−FR−Coworking:a1.1.1_indoor_plants

Types:

Aspect−FR−Coworking:Vegetal

Facts:

aspecto−fr:nominal_participant_in Aspect−FR−Coworking:e_biophilia,

aspecto−fr:s_directly_connected_to Aspect−FR−Coworking:a1.4_indoor_air

Individual: Aspect−FR−Coworking:a1.1_office_space

Types:

aspecto−fr:Technical_system

Facts:

DOLCE−Lite−FR:proper−part Aspect−FR−Coworking:a1.1.1_indoor_plants,

DOLCE−Lite−FR:proper−part Aspect−FR−Coworking:a1.2_open_kitchen,

DOLCE−Lite−FR:proper−part Aspect−FR−Coworking:a1.4_indoor_air,

DOLCE−Lite−FR:proper−part Aspect−FR−Coworking:a1.7_ping_pong_table,

DOLCE−Lite−FR:proper−part Aspect−FR−Coworking:a_daemon

Individual: Aspect−FR−Coworking:a1.2.1_boiler

Types:

Aspect−FR−Coworking:Boiler

Facts:

aspecto−fr:s_directly_connected_to Aspect−FR−Coworking:a1.4_indoor_air

Individual: Aspect−FR−Coworking:a1.2_open_kitchen

Types:

aspecto−fr:Technical_system

Facts:

aspecto−fr:s_directly_connected_to Aspect−FR−Coworking:a1.4_indoor_air,

DOLCE−Lite−FR:proper−part Aspect−FR−Coworking:a1.2.1_boiler

Individual: Aspect−FR−Coworking:a1.3.1_operable_vents

Individual: Aspect−FR−Coworking:a1.3_curtain_wall

Types:

aspecto−fr: Technical_artifact

Facts:

aspecto−fr:s_directly_connected_to Aspect−FR−Coworking:a0.1_outdoor_air,

aspecto−fr:s_directly_connected_to Aspect−FR−Coworking:a1.4_indoor_air,

DOLCE−Lite−FR:proper−part Aspect−FR−Coworking:a1.3.1_operable_vents

Individual: Aspect−FR−Coworking:a1.4_indoor_air

Types:

aspecto−fr:Air

Facts:

Aspect−FR−Coworking:relative_humidity 70,

Aspect−FR−Coworking:air_temperature 25

DifferentFrom:

Aspect−FR−Coworking:a0.1_outdoor_air

Individual: Aspect−FR−Coworking:a1.5_workers

Types:

Aspect−FR−Coworking:Animal

Facts:

aspecto−fr:s_directly_connected_to Aspect−FR−Coworking:a1.4_indoor_air

Individual: Aspect−FR−Coworking:a1.6_HVAC

Facts:

aspecto−fr:nominal_participant_in Aspect−FR−Coworking:e_ventilation

Individual: Aspect−FR−Coworking:a1.7_ping_pong_table

Facts:

DOLCE−Lite−FR:participant−in Aspect−FR−Coworking:e_recreation

Individual: Aspect−FR−Coworking:a1_office_building

Types:

aspecto−fr:Technical_system

Facts:

DOLCE−Lite−FR:proper−part Aspect−FR−Coworking:a1.1_office_space,

DOLCE−Lite−FR:proper−part Aspect−FR−Coworking:a1.3_curtain_wall,

DOLCE−Lite−FR:proper−part Aspect−FR−Coworking:a1.5_workers,

DOLCE−Lite−FR:proper−part Aspect−FR−Coworking:a1.6_HVAC

Individual: Aspect−FR−Coworking:a_daemon

Individual: Aspect−FR−Coworking:b_by_photosynthesis

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Coworking:e_transpiration

Individual: Aspect−FR−Coworking:b_condensation_capability

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Coworking:e_condensation

Individual: Aspect−FR−Coworking:b_maintain_IAQ_capability

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Coworking:e_indoor_air_quality

Individual: Aspect−FR−Coworking:b_moisture_exchange_capability

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Coworking:e_moisture_absortion

Individual: Aspect−FR−Coworking:b_moisture_release_capability

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Coworking:e_moisture_release

Individual: Aspect−FR−Coworking:b_productivity_capability

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Coworking:e_productivity

Individual: Aspect−FR−Coworking:b_transpiration_by_activity

Facts:

aspecto−fr:behavior_constraint_on_perdurant Aspect−FR−Coworking:e_transpiration

Individual: Aspect−FR−Coworking:e_air_contamination_removal

Individual: Aspect−FR−Coworking:e_biophilia

Individual: Aspect−FR−Coworking:e_boiling_water

Types:

Aspect−FR−Coworking:Boiling_water

Individual: Aspect−FR−Coworking:e_condensation

Facts:

aspecto−fr:_c_causes Aspect−FR−Coworking:e_mold_growth

Individual: Aspect−FR−Coworking:e_evaporation

Facts:

aspecto−fr:_c_causes Aspect−FR−Coworking:e_moisture_release

Individual: Aspect−FR−Coworking:e_fresh_air_exchange

Facts:

aspecto−fr:_c_causes Aspect−FR−Coworking:e_air_contamination_removal

Individual: Aspect−FR−Coworking:e_indoor_air_quality

Types:

aspecto−fr:OutPD

Facts:

DOLCE−Lite−FR:proper−part Aspect−FR−Coworking:e_air_contamination_removal,

DOLCE−Lite−FR:proper−part Aspect−FR−Coworking:e_spore_release

Individual: Aspect−FR−Coworking:e_moisture_absortion

Individual: Aspect−FR−Coworking:e_moisture_release

Facts:

aspecto−fr:_c_causes Aspect−FR−Coworking:e_moisture_absortion

Individual: Aspect−FR−Coworking:e_moisture_retention

Facts:

aspecto−fr:_c_causes Aspect−FR−Coworking:e_condensation

Individual: Aspect−FR−Coworking:e_moisture_transfer

Individual: Aspect−FR−Coworking:e_mold_growth

Facts:

aspecto−fr:_c_causes Aspect−FR−Coworking:e_spore_release

Individual: Aspect−FR−Coworking:e_nothing

Individual: Aspect−FR−Coworking:e_plant_metabolism

Types:

Aspect−FR−Coworking:Metabolism

Individual: Aspect−FR−Coworking:e_productivity

Facts:

DOLCE−Lite−FR:proper−part Aspect−FR−Coworking:e_biophilia,

DOLCE−Lite−FR:proper−part Aspect−FR−Coworking:e_indoor_air_quality,

DOLCE−Lite−FR:proper−part Aspect−FR−Coworking:e_recreation

Individual: Aspect−FR−Coworking:e_recreation

Individual: Aspect−FR−Coworking:e_spore_release

Individual: Aspect−FR−Coworking:e_transpiration

Facts:

aspecto−fr:_c_causes Aspect−FR−Coworking:e_moisture_release

Individual: Aspect−FR−Coworking:e_ventilation

Facts:

aspecto−fr:_c_causes Aspect−FR−Coworking:e_fresh_air_exchange

Individual: Aspect−FR−Coworking:fd_humidifier_daemon_function

Facts:

aspecto−fr:b0 Aspect−FR−Coworking:b_moisture_release_capability,

aspecto−fr:b1 Aspect−FR−Coworking:b_moisture_exchange_capability

Individual: Aspect−FR−Coworking:fe_failure_mode_condensation

Types:

aspecto−fr:_E_function

Facts:

aspecto−fr:_EF_in_environment <http://www.ou.edu/coa/ontologies/2018/6/Aspect−FR−KB#Env1>,

aspecto−fr:_EF_in_mode_of_deployment Aspect−FR−Coworking:m_md1,

aspecto−fr:_EF_nominal_artifact Aspect−FR−Coworking:a1.1_office_space,

aspecto−fr:b0 Aspect−FR−Coworking:fd_humidifier_daemon_function,

aspecto−fr:b1 Aspect−FR−Coworking:b_condensation_capability

Individual: Aspect−FR−Coworking:fe_indoor_air_quality

Types:

aspecto−fr:_E_function

Facts:

aspecto−fr:_EF_in_environment <http://www.ou.edu/coa/ontologies/2018/6/Aspect−FR−KB#Env1>,

aspecto−fr:_EF_nominal_artifact Aspect−FR−Coworking:a1.4_indoor_air,

aspecto−fr:b0 Aspect−FR−Coworking:b_condensation_capability,

aspecto−fr:b1 Aspect−FR−Coworking:b_maintain_IAQ_capability

Individual: Aspect−FR−Coworking:fe_productivity

Types:

aspecto−fr:_E_function

Facts:

aspecto−fr:_EF_in_environment <http://www.ou.edu/coa/ontologies/2018/6/Aspect−FR−KB#Env1>,

aspecto−fr:_EF_nominal_artifact Aspect−FR−Coworking:a1_office_building,

aspecto−fr:b0 Aspect−FR−Coworking:b_maintain_IAQ_capability,

aspecto−fr:b1 Aspect−FR−Coworking:b_productivity_capability

Individual: Aspect−FR−Coworking:m_md1

REFERENCES

[1] Aamodt, A. and Plaza, E., "Case-based reasoning - foundational issues, methodological variations, and system approaches," *Ai Communications*, vol. 7, no. 1, pp. 39–59, 1994.

[2] Akin, O., Sen, R., Donia, M., and Zhang, Y., "SEED-Pro: computer-assisted architectural programming in SEED," *Journal of Architectural Engineering*, vol. 1, no. 4, pp. 153–161, 1995.

[3] Akman, V., ten Hagen, P. J. W., and Tomiyama, T., "A fundamental and theoretical framework for an intelligent CAD system," *Computer-Aided Design*, vol. 22, no. 6, pp. 352–367, 1989.

[4] Al Horr, Y., Arif, M., Kaushik, A., Mazroei, A., Katafygiotou, M., and Elsarrag, E., "Occupant productivity and office indoor environment quality: A review of the literature," *Building and Environment*, vol. 105, pp. 369–389, 2016.

[5] Albano, L. D. and Suh, N. P., "Axiomatic design and concurrent engineering," *Computer-Aided Design*, vol. 26, no. 7, pp. 499–504, 1994.

[6] Albano, L. D. and Suh, N. P., "Axiomatic design and concurrent engineering," *Computer-Aided Design*, vol. 26, no. 7, pp. 499–504, 1994.

[7] Alexander, C., *A Pattern Language: Towns, Buildings, Construction.* Oxford University Press, 1977.

[8] Alexander, C., *A City is Not a Tree: 50th Anniversary Edition.* Sustasis Press, 2017.

[9] Alexander, C., *Notes on The Synthesis of Form.* United States: Harvard University Press, 1964.

[10] Allen, J. G., Macnaughton, P. D., Satish, U., Santanam, S., Vallarino, J. G., and Spengler, J. D., "Associations of cognitive function scores with carbon dioxide, ventilation, and volatile organic compound exposures in office workers: A controlled exposure study of green and conventional office environments," in *Environmental Health Perspectives*, pp. 805–812, 2016.

[11] Amor, R., Augenbroe, G., Hosking, J., Rombouts, W., and Grundy, j., "Directions in modelling environments," *Automation in Construction*, vol. 4, no. 3, pp. 173–187, 1995.

[12] Amor, R. and Faraj, I., "Misconceptions about integrated project databases," *Journal of information Technology in Construction*, vol. 6, no. 5, pp. 57–68, 2001.

[13] Arp, R. and Smith, B., "Function, role, and disposition in basic formal ontology," 2008.

[14] Arp, R., Smith, B., and Spear, A. D., *Building ontologies with basic formal ontology*. Cambridge, Massachusetts: The MIT Press, 2015.

[15] Ashley, K. D., "Reasoning with cases and hypotheticals in HYPO," *International Journal of Man-Machine Studies*, vol. 34, no. 6, pp. 753–796, 1991.

[16] Augenbroe, G., *The role of simulation in performance based building*, pp. 15–36. Spon Press, 2011.

[17] Augenbroe, G., "Integrated building performance evaluation in the early design stages," *Building and Environment*, vol. 27, no. 2, pp. 149–161, 1992.

[18] Augenbroe, G., "Trends in building simulation," *Building and Environment*, vol. 37, no. 8-9, pp. 891–902, 2002.

[19] Augenbroe, G., De Wilde, P., Moon, H. J., and Malkawi, A. M., "The Design Analysis Integration (DAI) Initiative," in *Eighth International IBPSA Conference*, pp. 79–86, 2003.

[20] Aurisicchio, M., Bracewell, R., and Armstrong, G., "The function analysis diagram: Intended benefits and coexistence with other functional models," *AI EDAM*, vol. 27, no. Special Issue 03, pp. 249–257, 2013.

[21] Bareiss, E. R., Porter, B. W., and Wier, C. C., "Protos: an exemplar-based learning apprentice," *International Journal of Man-Machine Studies*, vol. 29, no. 5, pp. 549–561, 1988.

[22] Bhatta, S. R. and Goel, A. K., "From design experiences to generic mechanisms: Model-based learning in analogical design," *Ai Edam-artificial Intelligence for Engineering Design Analysis and Manufacturing*, vol. 10, no. 2, pp. 294–300, 1996.

[23] B.J., K., "Commonsense reasoning about causality: deriving behavior from structure," *Artificial Intelligence*, vol. 24, pp. 169–203, 1984.

[24] BjKrórk, B.-C., "A conceptual model of spaces, space boundaries and enclosing structures," *Automation in Construction*, vol. 1, no. 3, pp. 193–214, 1992.

[25] BjKrórk, B.-C., "Requirements and information structures for building product data models," 1995-12-08.

[26] Bonwetsch, T., Bártschi, R., and Helmreich, M., *BrickDesign*, pp. 102–109. Vienna: Springer Vienna, 2013.

[27] Borgo, S., Carrara, M., Garbacz, P., and Vermaas, P. E., "A formal ontological perspective on the behaviors and functions of technical artifacts," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM*, vol. 23, pp. 3–21, 2009.

[28] Borgo, S. and Leitão, P., *Foundations for a Core Ontology of Manufacturing*, pp. 751–775. Boston, MA: Springer US, 2007.

[29] Borgo, S., Mizoguchi, R., and Smith, B., "On the ontology of functions," *Applied Ontology*, vol. 6, no. 2, pp. 99–104, 2011.

[30] Borgo, S., Sanfilippo, E. M., Šojić, A., and Terkaj, W., *Ontological Analysis and Engineering Standards: An Initial Study of IFC*, pp. 17–43. Springer International Publishing, 2015.

[31] Brown, D., "Functional, behavioral and structural features," in *Design Engineering Technical Conferences* (ASME, ed.), ASME, 2003.

[32] Brown, D. and Blessing, L., "The relationship between function and affordance," in *ASME 2005 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*, ASME, 2005.

[33] buildingSMART®, "Ifd library white paper," report, buildingSMART®, 2008.

[34] buildingSMART®, "IfcBuildingElement." Retrieved from http://www.buildingsmart-tech.org/ifc/IFC4/final/html/link/ifcbuildingelement.htm Last accessed on 07/24/2018., 2013.

[35] buildingSMART®, "IfcDistributionControlElement." Retrieved from http://www.buildingsmart-tech.org/ifc/IFC4/final/html/link/ifcdistributioncontrolelement.htm. Last accessed on 07/24/2018., 2013.

[36] buildingSMART®, "IfcDistributionFlowElement." Retrieved from http://www.buildingsmart-tech.org/ifc/IFC4/final/html/link/ifcdistributionflowelement.htm. Last accessed on 07/24/2018., 2013.

[37] buildingSMART®, "IfcEvent." Retrieved from http://www.buildingsmart-tech.org/ifc/IFC4/final/html/link/ifcevent.htm. Last accessed on 07/24/2018., 2013.

[38] buildingSMART®, "IfcGroup." Retrieved from http://www.buildingsmart-tech.org/ifc/IFC4/final/html/link/ifcgroup.htm. Last accessed on 07/24/2018, 2013.

[39] buildingSMART®, "IfcKernel." Retrieved from http://www.buildingsmart-tech.org/ifc/IFC4/final/html/schema/ifckernel/content.htm. Last accessed on 07/24/2018., 2013.

[40] buildingSMART®, "IfcObject." Retrieved from http://www.buildingsmart-tech.org/ifc/IFC4/final/html/link/ifcobject.htm. Last accessed on 07/24/2018., 2013.

[41] buildingSMART®, "IfcObjectDefinition." Retrieved from http://www.buildingsmart-tech.org/ifc/IFC4/final/html/link/ifcobjectdefinition.htm. Last accessed on 07/24/2018., 2013.

[42] buildingSMART®, "IfcProcedure." Retrieved from http://www.buildingsmart-tech.org/ifc/IFC4/final/html/link/ifcprocedure.htm. Last accessed on 07/24/2018., 2013.

[43] buildingSMART®, "IfcProcess." Retrieved from http://www.buildingsmart-tech.org/ifc/IFC4/final/html/link/ifcprocess.htm. Last accessed on 07/24/2018., 2013.

[44] buildingSMART®, "IfcPropertySingleValue." Retrieved from http://www.buildingsmart-tech.org/ifc/IFC4/final/html/link/ifcpropertysinglevalue.htm. Last accessed on 07/24/2018., 2013.

[45] buildingSMART®, "IfcProxy." Retrieved from http://www.buildingsmart-tech.org/ifc/IFC4/final/html/schema/ifckernel/lexical/ifcproxy.htm. Last accessed on 07/24/2018., 2013.

[46] buildingSMART®, "IfcRelDefinesByType." Retrieved from http://www.buildingsmart-tech.org/ifc/IFC4/final/html/link/ifcreldefinesbytype.htm. Last accessed on 07/24/2018., 2013.

[47] buildingSMART®, "IfcStructuralActivity." Retrieved from http://www.buildingsmart-tech.org/ifc/IFC4/final/html/link/ifcstructuralactivity.htm. Last accessed on 07/24/2018., 2013.

[48] buildingSMART®, "IfcSystem." Retrieved from http://www.buildingsmart-tech.org/ifc/IFC4/final/html/link/ifcsystem.htm. Last accessed on 07/24/2018, 2013.

[49] buildingSMART®, "IfcTypeObject." Retrieved from http://www.buildingsmart-tech.org/ifc/IFC4/final/html/link/ifctypeobject.htm. Last accessed on 07/24/2018., 2013.

[50] buildingSMART®, "Introduction." Retrieved from http://www.buildingsmart-tech.org/ifc/IFC4/Add2/html/introduction.htm. Last accessed on 07/24/2018., 2013.

[51] buildingSMART®, "buildingSMARTLinked Data Working Group." Retrieved from www.buildingsmart-tech.org/future/linked-data/. Last accessed on 07/24/2018., 2015.

[52] buildingSMART®, "IfcCurtainWall." Retrieved from http://www.buildingsmart-tech.org/ifc/IFC4/final/html/link/ifccurtainwall.htm. Last accessed on 07/24/2018., 2015.

[53] buildingSMART®, "IfcMaterial." Retrieved from http://www.buildingsmart-tech.org/ifc/IFC4/final/html/link/ifcmaterial.htm. Last accessed on 07/24/2018., 2015.

[54] buildingSMART®, "IfcResourceObjectSelect." Retrieved from http://www.buildingsmart-tech.org/ifc/IFC4/final/html/link/ifcresourceobjectselect.htm. Last accessed on 07/24/2018., 2015.

[55] buildingSMART®, "IfcWallTypeEnum." Retrieved from http://www.buildingsmart-tech.org/ifc/IFC4/final/html/link/ifcwalltypeenum.htm. Last accessed on 07/24/2018., 2015.

[56] buildingSMART®, "Pset_CurtainWallCommon." Retrieved from http://www.buildingsmart-tech.org/ifc/IFC4/final/html/link/pset_curtainwallcommon.htm. Last accessed on 07/24/2018., 2015.

[57] buildingSMART®, "Pset_WallCommon." Retrieved from http://www.buildingsmart-tech.org/ifc/IFC4/final/html/link/pset_wallcommon.htm. Last accessed on 07/24/2018., 2015.

[58] buildingSMART®, "History of buildingSMART." Retrieved from http://buildingsmart.org/about/about-buildingsmart/history/. Last accessed on 07/24/2018., 2016.

[59] buildingSMART®, "Standards overview page." Retrieved from http://buildingsmart.org/standards/. Last accessed on 07/24/2018., 2016.

[60] buildingSMART®- A Council of the National Institute of Building Sciences, "National BIM Standard - United States Version 3," report, National Institute of Building Sciences, 07/26/2016 2016.

[61] Bylander, T., "A theory of consolidation for reasoning about devices," *International Journal of Man-Machine Studies*, vol. 35, no. 4, pp. 467–489, 1991.

[62] Bylander, T. and Chandrasekaran, B., "Understanding behavior using consolidation," in *Proceedings of the 9th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'85, (San Francisco, CA, USA), pp. 450–454, Morgan Kaufmann Publishers Inc., 1985.

[63] C. Wolverton, B., Johnson, A., and Bounds, K., "Interior landscape plants for indoor air pollution abatement," report, National Aeronautics and Space Administration (NASA), 1989.

[64] Carbonell, J., "Derivational analogy : a theory of reconstructive problem solving and expertise acquisition," Report CMU-CS-85-115, Carnegie mellon University, 5 March 1985 1985.

[65] Carbonell, J., Etzioni, O., Gil, Y., Joseph, R., Knoblock, C., Minton, S., and Veloso, M., "PRODIGY: an integrated architecture for planning and learning," *SIGART Bulletin*, vol. 2, no. 4, pp. 51–55, 1991.

[66] Casati, R., Smith, B., and Varzi, A. C., *Ontological Tools for Geographic Representation*. Formal Ontology in Information Systems, Amsterdam, IOS Press, 1998.

[67] Cavieres, A., Al-Haddad, T., and Goodman, J., "Mounting clips for panel installation," 02/14/2017 2017.

[68] Cavieres, A. and Gentry, R., "Masonry regions: A new approach for the representation of masonry walls in bim applications," in *eCAADe 2015 - 33rd Annual Conference*, 2015.

[69] Cavieres, A., Gentry, R., and Al-Haddad, T., "Knowledge-based parametric tools for concrete masonry walls: Conceptual design and preliminary structural analysis," *Automation in Construction*, vol. 20, no. 6, pp. 716–728, 2011.

[70] Cavieres, A. and Goodman, J., "The role of functional requirements in multidisciplinary design: The case of application of photovoltaic systems to buildings," in *XVII Conference of the Iberoamerican Society pof Digital Graphics: Knowledge-based Design*, pp. 333–337, Editorial Universidad Técnica Federico Santa Maria, 2013.

[71] Chakrabarti, A. and Blessing, L., "Special issue: Representing functionality in design," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM*, vol. 10, no. 4, pp. 251–253, 1996.

[72] Chakrabarti, A., Sarkar, P., Leelavathamma, B., and Nataraju, B. S., "A functional representation for aiding biomimetic and artificial inspiration of new

ideas," *Artificial Intelligence in Engineering Design and Manufacturing*, vol. 19, no. 2, pp. 113–132, 2005.

[73] Chakrabarti, A., Srinivasan, V., Ranjan, B., and Lindemann, U., "A case for multiple views of function in design based on a common definition," *AI EDAM*, vol. 27, no. Special Issue 03, pp. 271–279, 2013.

[74] Chandrasekaran, B., Goel, A., and Iwasaki, Y., "Functional representation as design rationale," *Computer*, vol. 26, no. 1, pp. 48–56, 1993.

[75] Chandrasekaran, B. and Josephson, J., "Representing function as effect," in *Functional Modeling Workshop* (Modarres, M., ed.), (Paris, France), 1997.

[76] Chandrasekaran, B. and Josephson, J. R., "Function in device representation," *Engineering with Computers*, vol. 16, no. 3-4, pp. 162–177, 2000.

[77] Chaussecourte, P., Glimm, B., Horrocks, I., Motik, B., and Pierre, L., "The energy management adviser at EDF," in *Proc. of the 12th International Semantic Web Conference (ISWC 2013)*, Lecture Notes in Computer Science, 2013.

[78] Chen, Y., Zhang, Z., Huang, J., and Xie, Y., "Toward a scientific ontology based concept of function," *AI EDAM*, vol. 27, no. Special Issue 03, pp. 241–248, 2013.

[79] Chen, Y., Zhang, Z., Xie, Y., and Zhao, M., "A new model of conceptual design based on scientific ontology and intentionality theory. part i: The conceptual foundation," *Design Studies*, vol. 37, no. 0, pp. 12–36, 2015.

[80] Clayton, M., Teicholz, P., Fischer, M., and Kunz, J., "Virtual components consisting of form, function and behavior," *Automation in Construction*, vol. 8, no. 3, p. 16, 1999.

[81] Construction Industry Council, "Design Quality Indicator - DQI," 2015. Available online at http://www.dqi.org.uk/, last accessed on 12/18/2017.

[82] Construction Specification Canada and Construction Specification Institute, "OmniClass - Introduction and User's Guide," 2006.

[83] Construction Specification Institute, "OmniClass Standard," 2017.

[84] Coons, S., "An outline of the requirements for a computer-aided design system," in *AFIPS Spring Joint Computer Conference*, pp. 299–304, 1963.

[85] Crilly, N., "The roles that artefacts play: technical, social and aesthetic functions," *Design Studies*, vol. 31, no. 4, pp. 311–344, 2010.

[86] Crilly, N., "Function propagation through nested systems," *Design Studies*, vol. 34, no. 2, pp. 216–242, 2013.

[87] Crilly, N., "The proliferation of functions: Multiple systems playing multiple roles in multiple supersystems," *AI EDAM*, vol. 29, no. 01, pp. 83–92, 2015.

[88] Cross, N., "Expertise in design: an overview," *Design Studies*, vol. 25, no. 5, pp. 427–441, 2004.

[89] De Kleer, J., "Causal and teleological reasoning in circuit recognition," Technical Report AI-TR-529, Massachusetts Institute of Technology, 1979.

[90] De Kleer, J. and Brown, J. S., *A Qualitative Physics Based on Confluences*. Amsterdam: Elsevier, 1984.

[91] de Wilde, P., Augenbroe, G., and van der Voorden, M., "Design analysis integration: supporting the selection of energy saving building components," *Building and Environment*, vol. 37, no. 8-9, pp. 807–816, 2002.

[92] Delatte, N. J., *Beyond failure forensic case studies for civil engineers*. Reston, Virginia: American Society of Civil Engineers, 2009.

[93] Domeshek, E. A., Kolodner, J. L., and Zimring, C. M., *The Design Of A Tool Kit For Case-Based Design Aids*. Artificial Intelligence in Design '94, Springer, Dordrecht, 1994.

[94] Dorst, K., "The core of "design thinking" and its application," *Design Studies*, vol. 32, no. 6, pp. 521–532, 2011.

[95] Dorst, K. and Cross, N., "Creativity in the design process: co-evolution of problem-solution," *Design Studies*, vol. 22, no. 5, pp. 425–437, 2001.

[96] Drummond, N., Rector, A., Stevens, R., Moulton, G., Horridge, M., Wang, H., and Seidenberg, J., "Putting owl in order: Patterns for sequences in owl," in *OWLED*06 Workshop on OWL: Experiences and Directions*, 2006.

[97] Eastman, C. and Augenbroe, G., "Product modeling strategies for today and the future," in *CIB REPORT*, pp. 191–208, Netherlands, CIB CONSEIL INTERNATIONAL DU BATIMENT, 1998.

[98] Eastman, C., Lee, J.-m., Jeong, Y.-s., and Lee, J.-k., "Automatic rule-based checking of building designs," *Automation in Construction*, vol. 18, no. 8, pp. 1011–1033, 2009.

[99] Eastman, C. M., "Prototype integrated building model," *Computer-Aided Design*, vol. 12, no. 3, pp. 115–119, 1980.

[100] Eastman, C. M., Jeong, Y. S., Sacks, R., and Kaner, I., "Exchange model and exchange object concepts for implementation of National BIM Standards," *Journal of Computing in Civil Engineering*, vol. 24, no. 1, pp. 25–34, 2010.

[101] Eastman, C., "Automated space planning," *ARTIFICIAL INTELLIGENCE*, no. 4, pp. 41–64, 1970.

[102] Eastman, C., "The representation of design problems and maintenance of their structure," 1978.

[103] Eastman, C., "A data model for design knowledge," *Automation in Construction*, no. 3, pp. 135–147, 1994.

[104] Eastman, C., *New Directions in Design Cognition: Studies of Representation and Recall*, book section Chapter 8, pp. 147–198. Oxford: Elsevier Science, 2001.

[105] Eastman, C. and Henrion, M., "Glide: Glide. a lenguage for design information systems," 1977 1977.

[106] Eastman, C., Teicholz, P., Sacks, R., and Liston, K., *BIM handbook a guide to building information modeling for owners, managers, designers, engineers and contractors*. Wiley: Hoboken, NJ, 2nd ed. ed., 2011.

[107] Eastman, C. E., "GSP: A system for computer assisted space planning," 1971.

[108] Eastman, C. M., "Automated space planning," *Artificial Intelligence*, vol. 4, no. 1, pp. 41–64, 1973.

[109] Eastman, C. M., "Recent developments in representation in the science of design," *Design Studies*, vol. 3, no. 1, pp. 45–52, 1982.

[110] Eastman, C. M., "A data model analysis of modularity and extensibility in building databases," *Building and Environment*, vol. 27, no. 2, pp. 135–148, 1992.

[111] Eastman, C. M., "Modeling of buildings: evolution and concepts," *Automation in Construction*, vol. 1, no. 2, pp. 99–109, 1992.

[112] Eastman, C. M., *Building product models : computer environments supporting design and construction*. Boca Raton, Florida: CRC Press, 1999.

[113] Eastman, C. M., Chase, S. C., and Assal, H. H., "System architecture for computer integration of design and construction knowledge," *Automation in Construction*, vol. 2, no. 2, pp. 95–107, 1993.

[114] Eckert, C., "That which is not form: The practical challenges in using functional concepts in design," *AI EDAM*, vol. 27, no. Special Issue 03, pp. 217–231, 2013.

[115] Eisenbart, B., Gericke, K., and Blessing, L., "An analysis of functional modeling approaches across disciplines," *AI EDAM*, vol. 27, no. Special Issue 03, pp. 281–289, 2013.

[116] El-Bibany, H. E. and Paulson B.C, J., "A parametrics architecture for design, management, and coordination in a collaborative aec environment," *Computer-Aided Civil and Infrastructure Engineering*, vol. 14, no. 1, pp. 1–13, 1998.

[117] Elmasri, R. and S., N., *Database Systems*. Boston: Pearson Addison Wesley, 5th ed., 2006.

[118] Erden, M., Komoto, H., van Beek, T., D'Amelio, V., Echavarria, E., and Tomiyama, T., "A review of function modeling: Approaches and applications," *AI EDAM*, vol. 22, no. 02, pp. 147–169, 2008.

[119] Erhan, H. I., *Interactive Computational Support for Modeling and Generating Building Design Requirements*. Dissertation, School of Architecture, 2003.

[120] Fabi, V., Andersen, R. V., Corgnati, S., and Olesen, B. W., "Occupants' window opening behaviour: A literature review of factors influencing occupant behaviour and models," *Building and Environment*, vol. 58, pp. 188–198, 2012.

[121] Falkenhainer, B., "Analogy in context." Draft, 1990-.

[122] Falkenhainer, B., *A unified approach to explanation and theory formation*, book section Chapter 6, pp. 157–196. San Mateo, CA: Morgan Kaufmann, 1990.

[123] Falkenhainer, B. and Forbus, K. D., "Compositional modeling: finding the right model for the job," *Artificial Intelligence*, vol. 51, no. 1, pp. 95–143, 1991.

[124] Falkenhainer, B., F. K. and Gentner, D., "The structure-mapping engine," in *Proceedings of the Fifth AAAI National Conference on Artificial Intelligence*, pp. 272–277, AAAI, AAAI Press, 1986.

[125] Farias, T. M. d., Roxin, A., and Nicolle, C., "A rule-based methodology to extract building model views," *Automation in Construction*, vol. 92, pp. 214–229, 2018.

[126] Fenves, S. J., Rivard, H., and Gomez, N., "Seed-config: a tool for conceptual structural design in a collaborative building design environment," *Artificial Intelligence in Engineering*, vol. 14, no. 3, pp. 233–247, 2000.

[127] Fikes, R. E. and Nilsson, N. J., "Strips: A new approach to the application of theorem proving to problem solving," *Artificial Intelligence*, vol. 2, no. 3, pp. 189–208, 1971.

[128] Finkelstein, A. C. W., Gabbay, D., Hunter, A., Kramer, J., and Nuseibeh, B., "Inconsistency handling in multiperspective specifications," *Software Engineering, IEEE Transactions on*, vol. 20, no. 8, pp. 569–578, 1994.

[129] Fischer, G., "Turning breakdowns into opportunities for creativity," *Knowledge-Based Systems*, vol. 7, no. 4, pp. 221–232, 1994.

[130] Flager, F., Welle, B., Bansal, P., Soremekun, G., and Haymaker, J., "Multidisciplinary process integration and design optimization of a classroom building," *Journal of Information Technology in Construction*, vol. 14, pp. 595–612, 2009.

[131] Flemming, U. and Aygen, Z., "A hybrid representation of architectural precedents," *Automation in Construction*, vol. 10, no. 6, pp. 687–699, 2001.

[132] Flemming, U. and Woodbury, R., *Computer-Aided Design at the Department of Architecture Carnegie-Mellon Universit.* Pacifica, CA: Hurland/Swenson, 1985.

[133] Flemming, U. and Woodbury, R., "Software environment to support early phases in building design (SEED): Overview," *Journal of Architectural Engineering*, vol. 1, no. 4, pp. 147–152, 1995.

[134] Forbus, K., Stevens, A., Bolt, B., and Newman, *Using Qualitative Simulation to Generate Explanations.* Bolt Beranek and Newman, 1981.

[135] Franz, B., Diego, C., Deborah, L. M., Daniele, N., and Peter, F. P.-S., *The description logic handbook: theory, implementation, and applications.* Cambridge University Press, 2003.

[136] Freeman, P. and Newell, A., "A model for functional reasoning in design." Second International Joint Computer Conference on Artificial Intelligence, 1971.

[137] Friedenthal, S., Moore, A., and Steiner, R., *A Practical Guide to SysML - The Systems Modeling Language*. The MK/OMG Press, Morgan Kaufmann, 2011.

[138] Galle, P., "The ontology of Gero's FBS model of designing," *Design Studies*, vol. 30, no. 4, pp. 321–339, 2009.

[139] Gamma, E., *Design patterns : elements of reusable object-oriented software*. Reading, Mass.: Reading, Mass. : Addison-Wesley, 1995.

[140] Gangemi, A., "DOLCE-Lite." Retrieved from http://www.loa.istc.cnr.it/old/DOLCE.html/. Last accessed on 04/29/2018., 2006. Laboratory for Applied Ontology (LOA).

[141] Gentner, D., "Structure-mapping: A theoretical framework for analogy," *Cognitive Science*, vol. 7, no. 2, pp. 155–170, 1983.

[142] Gero, J. S., "Design prototypes: a knowledge representation schema for design," *AI Magazine*, vol. 11, no. 4, pp. 26–36, 1990.

[143] Geyer, P., "Component-oriented decomposition for multidisciplinary design optimization in building design," *Advanced Engineering Informatics*, vol. 23, no. 1, pp. 12–31, 2009.

[144] Geyer, P., "Systems modelling for sustainable building design," *Advanced Engineering Informatics*, vol. 26, no. 4, pp. 656–668, 2012.

[145] Geyer, P. and Buchholz, M., "Parametric systems modeling for sustainable energy and resource flows in buildings and their urban environment," *Automation in Construction*, vol. 22, pp. 70–80, 2012.

[146] Gibson, J. J., *The Theory of Affordances*. Lawrence Erlbaum Associates, Inc., 1979.

[147] Gielingh, W., "General AEC Reference Model," 1988. ISO TC184/SC4/WG1 document 3.2.2.1.

[148] Gilb, T. and Brodie, L., *Competitive engineering: a handbook for systems engineering, requirements engineering, and software engineering using Planguage*. Oxford ; Burlington, MA : Butterworth-Heinemann, 2005., 2005.

[149] Glimm, B., Horrocks, I., Motik, B., Stoilos, G., and Wang, Z., "HermiT: An OWL 2 Reasoner," *J. of Automated Reasoning*, vol. 53, no. 3, pp. 245–269, 2014.

[150] Goel, A., Bhatta, S., and Stroulia, E., *Kritik: An Early Case-Based Design System*, vol. 1, pp. 87–132. Lawrence Erlbaum Associates, 1997.

[151] Goel, A. and Chandrasekaran, B., "Functional representation of designs and redesign problem solving," in *Proceedings of the 11th International Joint Conference on Artificial Intelligence. Part 2 (of 2), Aug 20 - 25 1989*, vol. 2, pp. 1388–1394, Morgan Kaufmann Publ Inc, 1989.

[152] Goel, A. K., "Integrating case-based and model-based reasoning: a computational model of design problem solving," *AI Mag.*, vol. 13, no. 2, pp. 50–54, 1992.

[153] Goel, A. K., "Design, analogy, and creativity," *IEEE Expert Intelligent Systems & Their Applications*, vol. 12, no. 3, pp. 62–70, 1997.

[154] Goel, A. K., "A 30-year case study and 15 principles: Implications of an artificial intelligence methodology for functional modeling," *AI EDAM*, vol. 27, no. Special Issue 03, pp. 203–215, 2013.

[155] Goel, A. K. and Bhatta, S. R., "Use of design patterns in analogy-based design," *Ontology and it's Applications to Knowledge-Intensive Engineering*, vol. 18, pp. 85–94, 2004.

[156] Goel, A. K. and Chandrasekaran, B., *Case-Based Design: A Task Analysis*, book section 5, pp. 165–183. Boston: Academic Press, 1992.

[157] Goel, A. K., Rugaber, S., and Vattam, S. S., "Structure, behavior, and function of complex systems: The structure, behavior, and function modeling language," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM*, vol. 23, pp. 23–35, 2009.

[158] Goel, A. K. and Stroulia, E., "Functional device models and model-based diagnosis in adaptive design," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 10, no. 4, pp. 355–370, 1996.

[159] Goel, A. K., *Integration of case-based reasoning and model-based reasoning for adaptive design problem-solving*. Ph.d., Computer and Information Science, 1989.

[160] Goel, V. and Pirolli, P., "The structure of design problem spaces," *Cognitive Science*, vol. 16, no. 3, pp. 395–429, 1992.

[161] Goldschmidt, G., "Capturing indeterminism: representation in the design problem space," *Design Studies*, vol. 18, no. 4, pp. 441–455, 1997.

[162] Goodman, J., Porges, J., Lee, K., Cavieres, A., Fitzpatrick, P., and Al-Haddad, T., "Photovoltaic panel racking system," 2014.

[163] Grenon, P., "Bfo in a nutshell: A bi-categorial axiomatization of bfo and comparison with dolce," Report 06/2003, Faculty of Medicine, University of Leipzig, 2003.

[164] Group, O. M., "OMG Systems Modeling Language (OMG SysML)," June 2010 2010.

[165] Gruber, T. R., "A translation approach to portable ontology specifications," *Knowledge Acquisition.*, vol. 5, no. 2, pp. 199–220, 1993.

[166] Guarino, N., *Formal Ontology and Information Systems*. Formal Ontology in Information Systems, Trento, Italy: Amsterdam, IOS Press, 1998.

[167] Guarino, N., "Deliverable 1 general ontology of artefacts," report, Istituto di Scienze e Tecnologie della Cognizione del Consiglio Nazionale delle Ricerche, 2006.

[168] Guarino, N. and Borgo, S., "Deliverable 3 (final) general ontology of artefacts including subontology of information objects," report, Istituto di Scienze e Tecnologie della Cognizione del Consiglio Nazionale delle Ricerche, 2006.

[169] Haftka, R. T., Starnes Jr, J. H., Barton, F. W., and Dixon, S. C., "Comparison of two types of structural optimization procedures for flutter requirements," *AIAA Journal*, vol. 13, no. 10, pp. 1333–1339, 1975.

[170] Hammond, K. J., "CHEF: A model of case-based planning," 1986.

[171] Hayes, P. J., *The naive physics manifesto*, pp. 242–270. Edinburgh: Edinburgh University Press, 1979.

[172] Helms, M., *Analogical problem evolution in biologically inspired design*. Ph.d., School of Interactive Computing, 2013.

[173] Hertzberger, H., *Lessons for Students in Architecture*, vol. 1. Amsterdam: Uitgeverij010, 1991.

[174] Herzig, S. J. I., Qamar, A., and Paredis, C. J. J., "An approach to identifying inconsistencies in model-based systems engineering," *Procedia Computer Science*, vol. 28, no. 0, pp. 354–362, 2014.

[175] Herzig, S. J. I., Qamar, A., Reichwein, A., and Paredis, C. J. J., "A conceptual framework for consistency management in model-based systems engineering," in *2011 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, vol. 2, pp. 1329–1339, ASME, 2011.

[176] Hietanen, J., "IFC model view definition format," report, buildingSMART®, 2006.

[177] Hoes, P., Hensen, J. L. M., Loomans, M. G. L. C., de Vries, B., and Bourgeois, D., "User behavior in whole building simulation," *Energy and Buildings*, vol. 41, no. 3, pp. 295–302, 2009.

[178] Horridge, M. and Patel-Schneider, P. F., "Manchester syntax for owl 1.1," in *OWLED*, 2008.

[179] Horrocks, I., "What are ontologies good for?," in *Evolution of Semantic Systems* (Küppers, B.-O., Hahn, U., and Artmann, S., eds.), pp. 175–188, Springer, 2013.

[180] Horrocks, I. and Patel-Schneider, P. F., "KR and reasoning on the semantic web: OWL," in *Handbook of Semantic Web Technologies* (Domingue, J., Fensel, D., and Hendler, J. A., eds.), ch. 9, pp. 365–398, Springer, 2011.

[181] Horst, W. J. R. and Webber, M. M., "Dilemmas in a general theory of planning," *Policy Sciences*, no. 2, p. 155, 1973.

[182] Howard, T. J. and Andreasen, M. M., "Mind-sets of functional reasoning in engineering design," *AI EDAM*, vol. 27, no. Special Issue 03, pp. 233–240, 2013.

[183] Hua, K., Fairings, B., and Smith, I., "Cadre: case-based geometric design," *Artificial Intelligence in Engineering*, vol. 10, no. 2, pp. 171–183, 1996.

[184] Hull, E., Jackson, K., and Dick, J., *Requirements Engineering*. London : Springer-Verlag London Limited, 2011., 2011.

[185] Hwang, B. G., Thomas, S. R., Haas, C. T., and Caldas, C. H., "Measuring the impact of rework on construction cost performance," *Journal of Construction Engineering and Management*, vol. 135, no. 3, pp. 187–198, 2009.

[186] INCOSE, "Systems engineering vision 2020, version 2.03," report, International Council on Systems Engineering (INCOSE), 2007.

[187] International Organization for Standardization (ISO), "ISO 12006-3," 2007-04 2007.

[188] International Organization for Standardization (ISO), "ISO 16739:2013 Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries," 2013.

[189] International Organization for Standardization (ISO), "ISO 12006-2," 2015-05 2015.

[190] Ivan, E. S., "Sketchpad a man-machine graphical communication system," *SIMULATION*, vol. 2, no. 5, pp. R–3–R–20, 1964.

[191] Jin-Kook, L., Charles, M. E., Jaemin, L., Matti, K., and Yeon-Suk, J., "Computing walking distances within buildings using the universal circulation network," *Environment and Planning B: Planning and Design*, vol. 37, no. 4, pp. 628–645, 2010.

[192] Kalay, Y. E., *Architecture's new media : principles, theories, and methods of computer-aided design*. MIT Press: Cambridge, Mass., 2004.

[193] Kamara, J. M., Anumba, C. J., and Evbuomwan, N. F., "Capturing client requirements in construction projects," 2002.

[194] Kambhampati, S., Mali, A., and Srivastava, B., "Hybrid planning for partially hierarchical domains," in *AAAI-98 National Conference on Artificial Intelligence*, pp. 882–888, American Association for Artificial Intelligence, 1998.

[195] Karrbom Gustavsson, T. and Hallin, A., "Rethinking dichotomization: A critical perspective on the use of hard and soft in project management research," *International Journal of Project Management*, vol. 32, no. 4, pp. 568–577, 2014.

[196] Keeney, R. L., "Value-focused thinking: Identifying decision opportunities and creating alternatives," *European Journal of Operational Research*, vol. 92, no. 3, pp. 537–549, 1996.

[197] Kenneth D, F., "Qualitative process theory," *Artificial Intelligence*, vol. 24, no. 1-3, pp. 85–168, 1984.

[198] Keuneke, A. M., "Device representation-the significance of functional knowledge," *IEEE Expert: Intelligent Systems and Their Applications*, vol. 6, no. 2, pp. 22–25, 1991.

[199] Kitamura, Y., Koji, Y., and Mizoguchi, R., "An ontological model of device function: industrial deployment and lessons learned," *Applied Ontology*, vol. 1, no. 3/4, pp. 237–262, 2006.

[200] Kitamura, Y. and Mizoguchi, R., "Ontology-based systematization of functional knowledge," *Journal of Engineering Design*, vol. 15, no. 4, pp. 327–351, 2004.

[201] Kitamura, Y. and Mizoguchi, R., "Ontological characterization of functions: Perspectives for capturing functions and modeling guidelines," *AI EDAM*, vol. 27, no. Special Issue 03, pp. 259–269, 2013.

[202] Kiviniemi, A., *Requirements Management Interface to Building Product Models*. Dissertation, Department of Civil and Environmental Engineering", 2005.

[203] Kollia, I., Glimm, B., and Horrocks, I., "SPARQL query answering over OWL ontologies," in *Proc. of the 8th European Semantic Web Conf. (ESWC 2011)*, Lecture Notes in Computer Science, pp. 382–396, Springer, 2011.

[204] Kolodner, J. L., "Improving human decision-making through case-based decision aiding," *Ai Magazine*, vol. 12, no. 2, pp. 52–68, 1991.

[205] Kolodner, J., *Retrieval and organizational strategies in conceptual memory : a computer model*. Psychology Library Editions, Lawrence Erlbaum Associates, 1984.

[206] Kolodner, J., *Case-based reasoning*. Morgan Kaufmann Publishers Inc., 1993.

[207] Kolodner, J. L. and Riesbeck, C. K., *Experience, memory, and reasoning*. Hillsdale, N.J.: Hillsdale, N.J. : L. Erlbaum Associates, 1986.

[208] Krisnadhi, A., Maier, F., and Hitzler, P., *OWL and Rules*, pp. 382–415. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.

[209] Krótzsch, M., *OWL 2 Profiles: An Introduction to Lightweight Ontology Languages*, pp. 112–183. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

[210] Krótzsch, M., Simančík, F., and Horrocks, I., "Description Logics," *IEEE Intelligent Systems*, vol. 29, no. 1, pp. 12–19, 2014.

[211] Kuehn, T. H., *Thermal environmental engineering*. Upper Saddle River, N.J.: Upper Saddle River, N.J. : Prentice Hall, 3rd ed.. ed., 1998.

[212] Kurumatani, K., Tomiyama, T., and Yoshikawa, H., "Qualitative representation of machine behaviors for intelligent CAD systems," *Mechanism and Machine Theory*, vol. 25, no. 3, pp. 325–334, 1990.

[213] Larkin, J., McDermott, J., Simon, D. P., and Simon, H. A., "Expert and novice performance in solving physics problems," *Science*, vol. 208, no. 4450, pp. 1335–1342, 1980.

[214] Lê, M., Mohus, F., Kvarsvik, O., and Lie, M., "The HITOS project - a full scale IFC test," report, ECPPM, 2006.

[215] Leake, B. D., "Focusing construction and selection of abductive hypotheses," in *Proceedings of the 13th international joint conference on Artifical Intelligence - Volume 1*, (Chambery, France), pp. 24–29, Morgan Kaufmann Publishers Inc., 1993.

[216] Ledermann, C., Hanske, C., Wenzel, J., Ermanni, P., and Kelm, R., "Associative parametric CAE methods in the aircraft pre-design," *Aerospace Science and Technology*, no. 9, p. 10, 2005.

[217] Lee, B. D. and Paredis, C. J. J., "A conceptual framework for value-driven design and systems engineering," *Procedia CIRP*, vol. 21, pp. 10–17, 2014.

[218] Lee, G., Sacks, R., and Eastman, C., "Specifying parametric building object behavior (bob) for a building information modeling system," *Automation in Construction*, no. 15, pp. 758 – 776, 2005.

[219] Lee, J. M., *Automated checking of building requirements on circulation over a range of design phases*. Dissertation, School of Architecture, 2010.

[220] Lehmann, J., Borgo, S., Masolo, C., and Gangemi, A., *Causality and Causation in DOLCE*, vol. 114 of Frontiers in Artificial Intelligence and Applications, pp. 273–284. Amsterdam, IOS Press, 2004.

[221] Lipton, E., "Changes in design preceded collapse of casino garage," 2004.

[222] López, B. and Plaza, E., "Case-based learning of plans and goal states in medical diagnosis," *Artificial Intelligence in Medicine*, vol. 9, no. 1, pp. 29–60, 1997.

[223] Luth, G., "Chronology and context of the hyatt regency collapse," *Journal of Performance of Constructed Facilities*, vol. 14, no. 2, pp. 51–61, 2000.

[224] MacNaughton, P., Satish, U., Laurent, J. G. C., Flanigan, S., Vallarino, J., Coull, B., Spengler, J. D., and Allen, J. G., "The impact of working in a green certified building on cognitive function and health," *Building and Environment*, vol. 114, pp. 178–186, 2017.

[225] Maher, M. L. and Gomez de Silva Garza, A., "Developing case-based reasoning for structural design," *IEEE Expert*, vol. 11, no. 3, pp. 42–52, 1996.

[226] Maher, M. l., *CASECAD AND CADSYN: Implementing case retrieval and case adaptation*, vol. 1, book section 7, pp. 161–186. New York: Lawrence Erlbaum and Associates, 1997.

[227] Maher, M. L., Poon, J., and Boulanger, S., *Formalising Design Exploration As Co-Evolution: A Combined Gene Approach*, pp. 3–30. London: Chapman and Hall, London, 1996.

[228] Maher, M. L., Simoff, S. J., and Mitchell, J., "Formalising building requirements using an activity/space model," *Automation in Construction*, vol. 6, no. 2, pp. 77–95, 1997.

[229] Maier, J. R. A., Fadel, G. M., and Battisto, D. G., "An affordance-based approach to architectural theory, design, and practice," *Design Studies*, vol. 30, no. 4, pp. 393–414, 2009.

[230] Malak Jr, R. J. and Paredis, C. J. J., "Validating behavioral models for reuse," *Research in Engineering Design*, vol. 18, no. 3, pp. 111–128, 2007.

[231] Masolo, C., Borgo, S., Gangemi, A., Guarino, N., and Oltramari, A., "Wonderweb deliverable d18," report, Laboratory oor Applied Ontology - ISTC-CNR, 2003.

[232] Masonry Systems, "Cavity Wall: Brick Veneer/Steel Stud." Retrieved from http://www.masonrysystems.org/wall-systems/cavity-wall-brick-veneer-steel-stud/. Last accessed on 12/17/2017., 2015.

[233] McDermott, D., "Planning and acting," *Cognitive Science*, vol. 2, no. 2, pp. 71–109, 1978.

[234] Minsky, M., "A framework for representing knowledge," report, Massachusetts Institute of Technology, 1974.

[235] Mizoguchi, R., Kitamura, Y., and Borgo, S., "A unifying definition for artifact and biological functions," *Applied Ontology*, vol. 11, no. 2, pp. 129–154, 2016.

[236] Morgenstern, J., "The fifty-nine-story crisis," *THE NEW YORKER*, pp. 45–53, May 29, 1995 1995.

[237] Musen, M. A., "The protégé project: A look back and a look forward," *AI Matters*, vol. 1, no. 4, pp. 4–12, 2015.

[238] Newell, A. and Simon, H., *GPS, A Program That Simulates Human Thought*, pp. 109–124. Munchen: R. Oldenburg, 1961.

[239] Newell, A., S. J. C. and A., H. H., *Report on a General Problem-Solving Program*, pp. 256–264. UNESCO, 1959.

[240] Norberg-Schulz, C., *Genius loci: Towards a Phenomenology of Architecture*. Rizzoli, 1979.

[241] Ortin, F. and Cueva, J. M., "Dynamic adaptation of application aspects," *Journal of Systems and Software*, vol. 71, no. 3, pp. 229–243, 2004.

[242] Ortin, F., Labrador, M. A., and Redondo, J. M., "A hybrid class- and prototype-based object model to support language-neutral structural intercession," *Information and Software Technology*, vol. 56, no. 2, pp. 199–219, 2014.

[243] Oxman, R. E., "Precedents in design: a computational model for the organization of precedent knowledge," *Design Studies*, vol. 15, no. 2, pp. 141–157, 1994.

[244] Özkaya, I., *Requirement-driven design: assistance for information traceability in design computing*. Dissertation, School of Architecture, 2005.

[245] Ozkaya, I. and Akin, m., "Requirement-driven design: assistance for information traceability in design computing," *Design Studies*, vol. 27, no. 3, pp. 381–398, 2006.

[246] Ozkaya, I. and Akin, m., "Tool support for computer-aided requirement traceability in architectural design: The case of DesignTrack," *Automation in Construction*, vol. 16, no. 5, pp. 674–684, 2007.

[247] Pahl, G. and Beitz, W., *Engineering design : a systematic approach*. London: Springer, 1996.

[248] Panchal, J. H., Fernández, M. G., Paredis, C. J. J., Allen, J. K., and Mistree, F., "An interval-based constraint satisfaction (ibcs) method for decentralized, collaborative multifunctional design," *Concurrent Engineering Research and Applications*, vol. 15, no. 3, pp. 309–323, 2007.

[249] Pauwels, P., Van Deursen, D., Verstraeten, R., De Roo, J., De Meyer, R., Van de Walle, R., and Van Campenhout, J., "A semantic rule checking environment for building performance checking," *Automation in Construction*, vol. 20, no. 5, pp. 506–518, 2011.

[250] Pauwels, P., Törmä, S., Beetz, J., Weise, M., and Liebich, T., "Linked data in architecture and construction," *Automation in Construction*, vol. 57, pp. 175–177, 2015.

[251] Pearce, M., Goel, A. K., Kolodner, J. L., Zimring, C., Sentosa, L., and Billington, R., "Case-based design support: A case study in architectural design," *IEEE Expert: Intelligent Systems and Their Applications*, vol. 7, no. 5, pp. 14–20, 1992.

[252] Pfefferkorn, C. E., *The Design Problem Solver: A System for Designing Equipment or Furniture Layouts*, pp. 98–146. Wiley: New York, 1975.

[253] Phiri, M., *Information Technology in Construction Design*. London: Thomas Telford Publishing, 1999.

[254] Pohl, K., *Requirements Engineering: Fundamentals, Principles, and Techniques*. Berlin: Springer, 2010.

[255] Pollio, V., *Vitruvius: The Ten Books on Architecture*. Dover Publications: New York, 1960.

[256] Prabhakar, S. and Goel, A. K., "Functional modeling for enabling adaptive design of devices for new environments," *Artificial Intelligence in Engineering*, vol. 12, no. 4, pp. 417–444, 1998.

[257] Qamar, A., Paredis, C. J. J., Wikander, J., and During, C., "Dependency modeling and model management in mechatronic design," *Journal of Computing and Information Science in Engineering*, vol. 12, no. 4, pp. 041009–041009, 2012.

[258] Qian, L. and Gero, J., "A design support system using analogy," in *Second International Conference on Artificial Intelligence in Design* (Gero, J. S. and Sudweeks, F., eds.), pp. 795–813, Kluwer Academic Publishers, 1992.

[259] Raanaas, R. K., Evensen, K. H., Rich, D., Sjøstrøm, G., and Patil, G., "Benefits of indoor plants on attention capacity in an office setting," *Journal of Environmental Psychology*, vol. 31, no. 1, pp. 99–105, 2011.

[260] Rausand, M. and Øien, K., "The basic concepts of failure analysis," *Reliability Engineering & System Safety*, vol. 53, no. 1, pp. 73–83, 1996.

[261] Ren, Z., Yang, F., Bouchlaghem, N. M., and Anumba, C. J., "Multi-disciplinary collaborative building design - a comparative study between multi-agent systems and

multi-disciplinary optimisation approaches," *Automation in Construction*, vol. 20, no. 5, pp. 537–549, 2011.

[262] Riesbeck, C. K. and Schank, R. C., *Inside Case-Based Reasoning*. L. Erlbaum Associates Inc., 1989.

[263] Romani, Z., Draoui, A., and Allard, F., "Metamodeling the heating and cooling energy needs and simultaneous building envelope optimization for low energy building design in morocco," *Energy and Buildings*, vol. 102, pp. 139–148, 2015.

[264] Rush, R., *The Building Systems Integration Handbook*, vol. 1 of *The American Institute of Architects*. Stoneham, MA: Butterworth-Heinemann, 1985.

[265] Russell, S. J. and Norvig, P., *Artificial intelligence : a modern approach*. Prentice Hall series in artificial intelligence, Upper Saddle River, N.J. :: Prentice Hall/Pearson Education, 2nd ed. ed., 2003.

[266] Sacerdoti, E. D., "The nonlinear nature of plans," 1975.

[267] Sacks, R., Eastman, C., and Lee, G., "Parametric 3d modeling in building construction with examples from precast concrete," *Automation in Construction*, no. 13, pp. 291–312, 2003.

[268] Sadeghifam, A. N., Zahraee, S. M., Meynagh, M. M., and Kiani, I., "Combined use of design of experiment and dynamic building simulation in assessment of energy efficiency in tropical residential buildings," *Energy and Buildings*, vol. 86, pp. 525–533, 2015.

[269] Schamai, W., Helle, P., Fritzson, P., and Paredis, C. J., *Virtual Verification of System Designs against System Requirements*, vol. 6627 of *Lecture Notes in Computer Science*, book section 8, pp. 75–89. Springer Berlin Heidelberg, 2011.

[270] Schank, R. C., *Scripts, plans, goals, and understanding : an inquiry into human knowledge structures*. Hillsdale, N.J. :: L. Erlbaum Associates ;, 1977.

[271] Schank, R. C., *Dynamic Memory: A Theory of Reminding and Learning in Computers and People*. Cambridge University Press, 1983.

[272] Schank, R. C., *Explanation Patterns: Understanding Mechanically and Creatively*. L. Erlbaum Associates Inc., 1986.

[273] Schank, R. C., *Dynamic memory revisited*. Cambridge ;: Cambridge University Press, [2nd ed.] ed., 1999.

[274] Schank, R. C. and Abelson, R. P., "Scripts, plans, and knowledge," 1975.

[275] Schlueter, A. and Geyer, P., "Linking bim and design of experiments to balance architectural and technical design factors for energy performance," *Automation in Construction*, vol. 86, pp. 33–43, 2018.

[276] Schmit, L. A., "Structural synthesis - its genesis and development," *AIAA Journal*, vol. 19, no. 10, pp. 1249–1263, 1981.

[277] Searle, J. R., *The construction of social reality*. New York: Free Press, 1995.

[278] Sembugamoorthy, V. and Chandrasekaran, B., *Functional Representation of Devices and Compilation of Diagnostic Problem-Solving Systems*, book section 4, pp. 47–73. Hillsdale, N.J.: Lawrence Erlbaum Associate publishers, 1986.

[279] Sen, C. and Summers, J. D., "Identifying requirements for physics-based reasoning on function structure graphs," *AI EDAM*, vol. 27, no. Special Issue 03, pp. 291–299, 2013.

[280] Simon, H. A., *The sciences of the artificial*. MIT Press: Cambridge, Mass., 3rd ed. ed., 1996.

[281] Simon, H. A. and Newell, A., "Human problem solving: The state of the theory in 1970," *American Psychologist*, vol. 26, no. 2, pp. 145–159, 1971.

[282] Simons, P., *Parts: A Study in Ontology*. New York: Clarendon Press, Oxford, 1987.

[283] Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A., and Katz, Y., "Pellet: A practical owl-dl reasoner," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 5, no. 2, pp. 51–53, 2007.

[284] Smith, B. and Mark, D. M., "Do mountains exist? towards an ontology of landforms," *Environment and Planning B: Planning and Design*, vol. 30, no. 3, pp. 411–427, 2003.

[285] Smith, B., "Beyond concepts: Ontology as reality representation," in *Proceedings of the Third International Conference on Formal Ontology in Information Systems (FOIS-2004)* (Varzi, A. C. and Vieu, L., eds.), pp. 73–84, IOS Press, 2004.

[286] Smith, B., *Against Fantology*. Experience and Analysis, OBV & HPT, 2005.

[287] Smith, B., "Against idiosyncrasy in ontology development," in *Proceedings of the Fourth International Conference on Formal Ontology in Information Systems (FOIS-2006)* (Bennett, B., F. C., ed.), pp. 15–26, IOS Press, 2006.

[288] Smith, B., "Classifying processes: An essay in applied ontology," *Ratio*, vol. 25, no. 4, pp. 463–488, 2012.

[289] Smith, B., Ashburner, M., Rosse, C., Bard, J., Bug, W., Ceusters, W., Goldberg, L. J., Eilbeck, K., Ireland, A., Mungall, C. J., Leontis, N., Rocca-Serra, P., Ruttenberg, A., Sansone, S.-A., Scheuermann, R. H., Shah, N., Whetzel, P. L., and Lewis, S., "The obo foundry: coordinated evolution of ontologies to support biomedical data integration," *Nat Biotech*, vol. 25, no. 11, pp. 1251–1255, 2007.

[290] Smith, B. and Grenon, P., "The cornucopia of formal-ontological relations," *Dialectica*, vol. 58, no. 3, pp. 279–296, 2004.

[291] Solibri. Retrieved from https://www.youtube.com/watch?time_continue=67&v=EOJgVOD-fzU. Last accessed on 07/24/2018., 12/18/2017 2014.

[292] Solihin, W. and Eastman, C., "Classification of rules for automated bim rule checking development," *Automation in Construction*, vol. 53, no. Supplement C, pp. 69–82, 2015.

[293] Sowa, J. F. and Majumdar, A. K., "Analogical reasoning," in *Conceptual Structures for Knowledge Creation and Communication*, pp. 16–36, Springer Berlin Heidelberg, 2003.

[294] Stewart, C. and Lee, K., *Computer-Aided Programming and Schematic Design of Health Facilities*, pp. 60–79. Boston :: Environ Design, 1973.

[295] Stone, R. B. and Wood, K. L., "Development of a functional basis for design," *Journal of Mechanical Design*, vol. 122, no. 4, pp. 359–370, 1999.

[296] Succar, B., "Building information modelling framework: A research and delivery foundation for industry stakeholders," *Automation in Construction*, vol. 18, no. 3, pp. 357–375, 2009.

[297] Suh, N. P., *The principles of design*. Oxford series on advanced manufacturing ;, New York :: Oxford University Press, 1990.

[298] Suh, N. P., "Axiomatic design theory for systems," *Research in Engineering Design*, vol. 10, no. 4, pp. 189–209, 1998.

[299] Suh, N. P., *Axiomatic design : advances and applications*. The MIT-Pappalardo series in mechanical engineering, New York :: Oxford University Press, 2001.

[300] Tang, D., Zhang, G., and Dai, S., "Design as integration of axiomatic design and design structure matrix," *Robotics and Computer-Integrated Manufacturing*, vol. 25, no. 3, pp. 610–619, 2009.

[301] Tobies, S., "The complexity of reasoning with cardinality restrictions and nominals in expressive description logics," *Journal of Artificial Intelligence Research*, vol. 12, pp. 199–217, 2000.

[302] Tolk, A. and Smith, B., *Command and Control Ontology*, vol. 4. 2011.

[303] Tomiyama, T. and ten Hagen, P. J. W., "Representing knowledge in two distinct descriptions: Extensional vs. intensional," *Artificial Intelligence in Engineering*, vol. 5, no. 1, pp. 23–32, 1990.

[304] Tomiyama, T., van Beek, T. J., Alvarez Cabrera, A. A., Komoto, H., and D'Amelio, V., "Making function modeling practically usable," *AI EDAM*, vol. 27, no. Special Issue 03, pp. 301–309, 2013.

[305] Tsai, W.-J. J., *A Pragmatic Approach in Supporting Multidisciplinary Communication and Negotiation in Building Design*. Dissertation, School of Architecture, 2005.

[306] Tsarkov, D., Riazanov, A., Bechhofer, S., and Horrocks, I., "Using vampire to reason with owl," in *Lecture Notes in Computer Science book series (LNCS, volume 3298)* (McIlraith, S. A., Plexousakis, D., and van Harmelen, F., eds.), pp. 471–485, Springer Berlin Heidelberg, 2004.

[307] Ulrich, R. S., Zimring, C., Zhu, X., DuBose, J., Seo, H.-B., Choi, Y.-S., Quan, X., and Joseph, A., "A review of the research literature on evidence-based healthcare design," *HERD: Health Environments Research & Design Journal*, vol. 1, no. 3, pp. 61–125, 2008.

[308] Umeda, Y. and Tomiyama, T., "Functional reasoning in design," *IEEE Expert: Intelligent Systems and Their Applications*, vol. 12, no. 2, pp. 42–48, 1997.

[309] Varanka, D. and Usery, E., *An Applied Ontology for Semantics Associated with Surface Water Features*. 2015.

[310] Vattam, S., Wiltgen, B., Helms, M., Goel, A. K., and Yen, J., "Dane: Fostering creativity in and through biologically inspired design," in *Design Creativity 2010*, pp. 115–122, Springer London, 2010.

[311] Veloso, M., Carbonell, J., Pérez, A., Borrajo, D., Fink, E., and Blythe, J. I. M., "Integrating planning and learning: the PRODIGY architecture," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 7, no. 1, pp. 81–120, 1995.

[312] Venugopal, M., *Formal Specification of Industry Foundation Class Concepts using Engineering Ontologies*. Ph.d., School of Civil and Environmental Engineering, 2011.

[313] Venugopal, M., Eastman, C., and J., T., "Formal specification of the IFC concept structure for precast model exchanges," in *Computing in Civil Engineering* (Issa, R. R. and Flood, I., eds.), pp. 213–220, ASCE, 2012.

[314] Vermaas, P. E., "On the formal impossibility of analysing subfunctions as parts of functions in design methodology," *Research in Engineering Design*, vol. 24, no. 1, pp. 19–32, 2013.

[315] Vermaas, P. E., "The coexistence of engineering meanings of function: Four responses and their methodological implications," *AI EDAM*, vol. 27, no. Special Issue 03, pp. 191–202, 2013.

[316] Vermaas, P. E., "The conceptual elusiveness of engineering functions: A philosophical analysis," *Philosophy and Technology*, vol. 26, pp. 159–185, 2013.

[317] Vermaas, P. E. and Dorst, K., "On the conceptual framework of john gero's fbs-model and the prescriptive aims of design methodology," *Design Studies*, vol. 28, no. 2, pp. 133–157, 2007.

[318] Vermaas, P. E. and Eckert, C., "My functional description is better!," *AI EDAM*, vol. 27, no. Special Issue 03, pp. 187–190, 2013.

[319] Voss, A., *Case Design Specialists in FABEL*, vol. 1, book section 12. New York: Lawrence Erlbaum Associates, 1997.

[320] Walker, W. E., Harremoës, P., Rotmans, J., van der Sluijs, J. P., van Asselt, M. B. A., Janssen, P., and Krayer von Krauss, M. P., "Defining uncertainty: A conceptual basis for uncertainty management in model-based decision support," *Integrated Assessment*, vol. 4, no. 1, pp. 5–17, 2003.

[321] Wang, L. and Leite, F., "Formalized knowledge representation for spatial conflict coordination of mechanical, electrical and plumbing (mep) systems in new building projects," *Automation in Construction*, vol. 64, no. Supplement C, pp. 20–26, 2016.

[322] Wang, L. and Leite, F., "Process knowledge capture in bim-based mechanical, electrical, and plumbing design coordination meetings," *Journal of Computing in Civil Engineering*, vol. 30, no. 2, p. 04015017, 2016.

[323] Wang, L., Shen, W., Xie, H., Neelamkavil, J., and Pardasani, A., "Collaborative conceptual design–state of the art and future trends," *Computer-Aided Design*, vol. 34, no. 13, pp. 981–996, 2002.

[324] Weinzapfel, G. and Handel, S., *IMAGE: Computer Assistant for Architectural Design*, pp. 61–97. Wiley: New York, 1975.

[325] Wilensky, R., *Planning and understanding: A computational approach to human reasoning*. Addison-Wesley Pub. Co.,Reading, MA; None, 1983.

[326] Wix, J. and Karlshøj, J., "Information delivery manual guide to components and development methods," report, buildingSMART®, 2010.

[327] Wölkl, S. and Shea, K., "A computational product model for conceptual design using sysml," in *Proceedings of the ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference 2009*, vol. 2, pp. 635–645, 2009.

[328] Wood, W. H. and Agogino, A. M., "Case-based conceptual design information server for concurrent engineering," *Computer-Aided Design*, vol. 28, no. 5, pp. 361–369, 1996.

[329] World Wide Web Consortium (W3C), "Defining n-ary relations on the semantic web," report, W3C, 2006.

[330] Yaner, P. W. and Goel, A. K., "Analogical recognition of shape and structure in design drawings," *AI EDAM*, vol. 22, no. 02, pp. 117–128, 2008.

[331] Yurchyshyna, A. and Zarli, A., "An ontology-based approach for formalisation and semantic organisation of conformance requirements in construction," *Automation in Construction*, vol. 18, no. 8, pp. 1084–1098, 2009.

[332] Zhang, S., Teizer, J., Lee, J.-K., Eastman, C. M., and Venugopal, M., "Building information modeling (bim) and safety: Automatic safety checking of construction models and schedules," *Automation in Construction*, vol. 29, no. 0, pp. 183–195, 2013.

[333] Zimring, C., Augenbroe, G. L., Malone, E. B., and Sadler, B. L., "Implementing healthcare excellence: The vital role of the ceo in evidence-based design," *HERD: Health Environments Research & Design Journal*, vol. 1, no. 3, pp. 7–21, 2008.

[334] Zolin, E., "Complexity of reasoning in description logics." Retrieved from http://www.cs.man.ac.uk/~ezolin/dl/. Last accessed on 07/24/2018., 2013.

VITA

Andrés Cavieres received his Bachelor and Professional degrees in Architecture from the Facultad de Arquitectura y Urbanismo de la University of Chile. Immediately after graduation, Andres became a lecturer in the same college at Universidad de Chile, where he became involved in teaching and research on topics related to Design Cognition and Computation and Knowledge Representation in Design.

During the course of his Ph.D. studies at Georgia Tech, Andres was part of the Digital Fabrication Lab (DFL) and Digital Building Lab (DBL), working on the specification Building Information Modeling applications for the masonry industry (BIM-M initiative). He was also a researcher at the Georgia Tech Research Institute, working on a multi-year interdisciplinary research effort funded by the U.S. Department of Energy, where he led the development of several patented technologies for solar energy applications.

While still at Georgia Tech, Andrés was awarded the King Student Medal for Excellence in Architectural + Environmental Research, by the Architectural Research Centers Consortium - ARCC. His work has been published in journals and conference proceedings, and exhibited at the Biennale of Architecture in São Paulo, the Museum of Design of Atlanta; at the AIA New York Center for Architecture. Currently, he is an Assistant Professor at the Christopher C. Gibbs College of Architecture, at the University of Oklahoma.